(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number
WO 01/06805 A1

(51) International Patent Classification[7]: H04Q 7/38, G06F 1/00

(21) International Application Number: PCT/SE00/01418

(22) International Filing Date: 4 July 2000 (04.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
9902746-8 16 July 1999 (16.07.1999) SE

(71) Applicant: TELEFONAKTIEBOLAGET LM ERICSSON (publ) [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventors: HÅKANS, Anders; S:t Eriksgatan 106, S-113 31 Stockholm (SE). REICHERT, Frank; Shangri-La Residences #01-09, Lady Hill Rd. 1A, 258685 Singapore (SG).

(74) Agent: ERICSSON MOBILE COMMUNICATIONS AB; Patent Unit, S-164 80 Stockholm (SE).

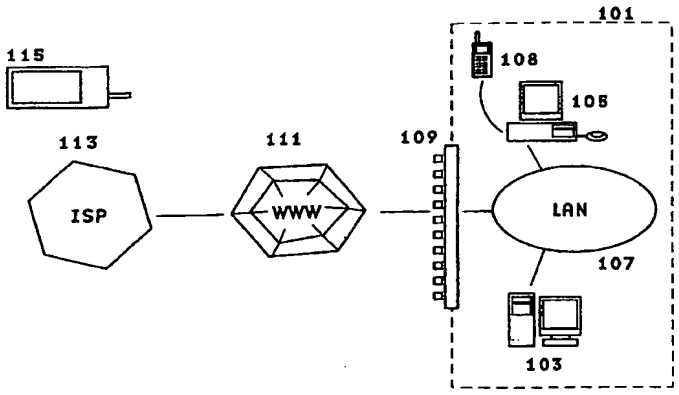
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*
— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD AND A SYSTEM FOR AUTHENTICATING A MOBILE COMMUNICATION DEVICE



(57) Abstract: In a method and a system for providing an authentication of a mobile communication device (115) a code word is generated in a computer system (101) which code word is transmitted to the device (115). When receiving the code word the device automatically generates a code message using the code word, and transmits the code message to the computer system. Data is then transmitted from the computer system to the device if the code message is authenticated by the computer system. In a preferred embodiment the device useS a Hypertext Transfer Protocol (HTTP) or Simple Mail Transport Protocol (SMTP) to transmit the code message. This is advantageous since a firewall is designed to let traffic using such protocols through. Thus, the security provided by the firewall is not reduced. By using such a method and system, several different types of authenticated data transmission can be obtained. For example, automated forwarding of data, such as electronic mail (E-mail), facilitated logging in on a secure network, such as a bank and many more applications where an authentication of a user is required or desired.

WO 01/06805 A1

A METHOD AND A SYSTEM FOR AUTHENTICATING A MOBILE COMMUNICATION DEVICE

TECHNICAL FIELD

The present invention relates to a method and a system for authenticating a mobile communication device. In particular the invention relates to a transmission system, which transmits data between a computer system and a mobile communication device, and to a method for authenticating a mobile communication device to a computer system.

BACKGROUND OF THE INVENTION AND PRIOR ART

In many data communication applications there is a need for a data transmission between a computer system located inside a firewall and a mobile client located outside the firewall. An example of such an application is when a client or user connects to a bank in order to carry out a financial transaction. In such a case it is of course important that the user can authenticate himself, in order to provide a secure financial transaction system. Another example of such an application is for forwarding messages from a company to an employee. It is in such a case important that the information is forwarded in a secure way and to the right address.

It is thus important to provide a secure method of authenticating a client. However, it is also desired that the authentication process is user friendly.

Also, in the International patent application WO 97/08906 a system and host arrangement for transmission of electronic mail is described. The arrangement uses short text message (SMS) for transmitting an electronic mail (E-mail) to a mobile station.

SUMMARY

It is an object of the present invention to provide a method and a system for authenticating a mobile telecommunication device, and transmit data from a computer system to an authenticated mobile telecommunication device or from an authenticated device to a computer system.

This object and others are obtained by a method and a system arranged to carry out the following steps:

(I) - Generating a code word in the computer system, which code word is transmitted to the device.

(II) - Receiving the code word in the device,

(III) - Generating a code message, in the device, based on the received code word, the generation being performed in a pre-defined manner.

(IV) - Transmitting the code message from the device to the computer system,

(V) – Using the code word to check the code message, and

(VI) - Transmitting data from the computer system to the device if the code message is verified by the computer system.

Prior to step (V) the computer system transmits information (e.g. the code word and a "you've got mail"-message) used in the authentication process, but the device has not access to data in the computer system. Preferably the computer system generates a random code word. Preferably, when the computer system transmits the code word to the device in step (I), the computer system uses a predetermined address of the device. Furthermore, the generation of the code message, in step (III) is preferably performed by encryption of the received code word using a unique encryption key of the device. If the code message is verified, step (VI), then the address of the device is authenticated for data transmission.

In a preferred embodiment the device transmits the code message using a Hypertext Transfer Protocol (HTTP) or Simple Mail Transport Protocol (SMTP). This is advantageous since a firewall is designed to let traffic using such protocols through. In other words the security provided by the firewall is not reduced. Also preferably, the computer system transmits data in step (VI) to the address from where the device sent the code message, e.g. the computer system extracts the sender address of the mobile communication device and transmits the data to this address. This address can differ from the address to which the code word was transmitted.

By using such a method and system, several different types of authenticated data transmission can be obtained. For example, automated forwarding of data, such as electronic mail (E-mail), facilitated logging in on a secure network, such as a bank and many more applications where an authenticated transmission is required or desired.

The use of the system and the method as described herein also facilitates the authentication from a user point of view. Thus, there is no need for a one time password generator at the client side of the system, since the random code word is generated at the computer system side of the system.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described in more detail by way of non-limiting examples and with reference to the accompanying drawings, in which:

- Fig. 1 is a schematic view of a system for automatic forwarding of E-mail or other data using the Internet.
- Fig. 2 is a schematic view of a system for automatic forwarding of E-mail or other data using a packet data network.
- Fig. 3 is a schematic view of a system for automatic authentication of a device over an Internet connection.
- Fig. 4 is a flow chart illustrating the steps carried out when authenticating a mobile communication device and initiating a data transmission in the system shown in Fig. 1.
- Figs. 5a and 5b are flow charts illustrating the steps carried out when authenticating a mobile communication device and initiating a data transmission in the system shown in Fig. 2.
- Fig. 6 is a flow chart illustrating the steps carried out when authenticating a mobile communication device in the system shown in Fig. 3.

DESCRIPTION OF PREFERRED EMBODIMENTS

In Fig. 1, a schematic view of a system for automatic forwarding of E-mail or other data using an Internet connection is shown. The system comprises a computer system 101, which in turn can comprise a number of different computers and other devices. The computer system 101 in Fig. 1 comprises an electronic mail (E-mail) server 103 connected to a local area network (LAN) 107 of the computer system 101. The server 103 is arranged to handle electronic mails (E-mails) within the computer system 101. The computer system 101 also comprises an authentication server 105, which also is connected to the LAN 107 and to a communication device 108 for transmitting messages outside the computer system 101.

The entire computer system 101 is, for example, located inside a firewall 109. The only way for entering data from outside the firewall 109 to the computer system 101 is through the firewall 109. The firewall 109 is arranged to only let trough data traffic arriving in a HTTP or SMTP format.

A mobile communication device 115 is located outside the firewall. The firewall is connected to the Internet 111, which in turn is connected to a number of Internet service providers (ISP) 113. The Internet service providers 113 provides an interface via which different devices, such as the mobile communication device 115, can connect to the Internet.

In Fig. 4, a flow chart illustrating different steps carried out when authenticating a mobile communication device 115 and initiating a data transmission from the computer system 101 to the mobile communication device 115 in the system shown in Fig. 1.

First some event, such as the arrival of an E-mail to the E-mail server 103 of the computer system 101, which E-mail is addressed to a user of the device 115, triggers the set up of an authenticated transmission towards the device 115 from the computer system 101, step 401. I.e. the computer system 101 is arranged to automatically forward data, in this case E-mails, which is present inside the computer system and which is intended for a client which presently is located outside the computer system 101 and hence also outside the firewall 109.

The system 101 then generates a random code word, step 403. The code word is generated by the authentication server 105. The code word generated by the authentication server 105 is then transmitted via the communication device 108, for example by means of a transmitting a short message service (SMS) message, to the device 115, step 405. The authentication server uses a predetermined address of the user, in this case an SMS number.
The device 115 receives the code word from the computer system 101, step 407. In response to this reception the device automatically connects to an Internet service provider (ISP) 113, step 409. When the device 115 connects to the ISP 113 it receives an Internet protocol (IP) address from the ISP 113, step 411. The device also generates a code message based on the received code word, step 414.

When the device has received an IP address, an E-mail server is set up inside the device, step 413. An E-mail is then transmitted to the authentication server 105 of the computer system, step 415. The E-mail from the device 115 to the authentication server 105 is transmitted via the ISP 113, the Internet 111, through the firewall 109, and via the LAN 107.

The E-mail from the device 115 to the authentication server 105 comprises at least the code message. The code message is preferably generated by encrypting the received code word using a unique encryption key. In particular the code message is generated by encrypting the code word together with a part of the encryption key.

In a preferred embodiment the E-mail from the device to the authentication server 105 is transmitted using a SMTP or HTTP protocol, since the firewall 109 is designed to only let such data traffic through. Thus, the firewall 109 can maintain a high level of security. Preferably the SMTP protocol is used, since it only requires one socket to be used for the connection.

Thereupon, the authentication server receives the code message from the device, step 417. The code word is then extracted from the received E-mail, which may require decryption if the code word is encrypted, step 419. Next, it is checked if the transmitted code word matches the extracted code word, step 421.

If the extracted code word does not match the transmitted code word, the session ends, step 423. However, in a preferred embodiment the computer system 101 can be arranged to continue to try to establish an authenticated connection to the device 115 later.

If the code word, when extracted, matches the code word originally transmitted to the device 115, the authentication server knows that it is the correct device.

Now the authentication server knows where to transmit the data, in this example an E-mail. Thus, the authentication server 105 copies the data from the E-mail server and transmits the E-mail to the device 115 using the IP-address of the device, step 425. The data transmitted is preferably encrypted using the unique encryption key of the device 115.

In Fig. 2 another system for automatic forwarding of E-mail or other data, which uses packet data transmission is shown. The system as shown in Fig. 2 is similar to the system shown in Fig. 1. However, instead of using an ISP 113 for transmitting data between the computer system 101 and the device 115, a packet data network 112 is used. Of course, the Internet can be used for transmissions between the computer system 101 and the packet data network 112.

In Fig. 5a, a flow chart illustrating the steps carried out when authenticating a mobile communication device in the system shown in Fig. 2 is shown. The flow chart in Fig. 5 is similar to the flow chart in Fig. 4. However, the transmission from the device 115 to the computer system 101 is done via the packet data network, 112.

Also, since in this case the device already is connected to the mobile network, when the code word is transmitted, the code word can be transmitted to a predetermined address of the device using the packet data network. The code word can then be transmitted embedded in a data packet.

In Fig. 5b, a flow chart illustrating yet another way of authenticating a mobile communication device is shown, which is similar to the method described above in conjunction with Fig. 5a.

The difference between the flow chart in Fig. 5a and the flow chart in Fig. 5b is that the authentication server extracts the code word (step 419 in Fig. 5a), and compares it to the code

word; and that the authentication server extracts the code message (step 418 in Fig. 5b), and compares it to a code message generated by itself using the same code message generating algorithm as the device.

Thus, upon reception of the code word, the device generates a code message using the code word as input data in a code message generating algorithm, step 414. The system then uses the same code message generation algorithm as the device, step 420 in order to authenticate the device. The device generates the code message, step 414, during the time interval between the reception of the code word, step 407, and the transmission of the E-mail, step 415. The authentication server generates the code message, step 420 during the time interval between the generation of the code word, step 403, and the comparison of the code messages, step 421.

In Fig. 3 a view of a system for automatic authentication of a device over an Internet connection is shown. The system used can be similar to the system as shown in Fig. 1 or Fig. 2. The system as shown in Fig. 3 also comprises a financial transaction server 104, which is connected to the LAN 107.

In Fig. 6 a flow chart illustrating the steps carried out when authenticating the device 115 to the computer system 101 in the system shown in Fig. 3 is shown.

Thus, when a user of the device 115 wants to be authenticated for some reason, for example he/she may want to carry out financial transactions in the financial transaction server 104 using a secure connection, the user transmits an E-mail from the device 115 to the authentication server 105 requesting to be authenticated, step 400. The E-mail, which comprises information identifying the user, such as a user ID, is transmitted via the ISP 113, the Internet 111, through the firewall 109, and via the LAN 107 to the authentication server 105.

The computer system 101 receives the request, step 402. In response to this request the authentication server 105 generates a code word, in particular a random code word, step 403. The system 101 then transmits the code word via the communication device 108, for example by means of transmitting a short message service (SMS) message, to the device 115, step 405. The computer system sends the code word to a predetermined address of the device 115. The computer system 101 uses the received identification information from the user together with a stored address list to obtain the (predetermined) address. In another preferred embodiment the random code word is transmitted on the Internet to a server or the like, which server in turn transmits an SMS to the device 115.

The device 115 receives the code word from the computer system 101, step 407, and generates a code message. In response to this reception the device automatically transmits a second E-mail, including the code message, to the authentication server 105 of the computer system, step 415. The second E-mail from the device 115 to the authentication server 105 is transmitted via the ISP 113, the Internet 111, through the firewall 109, and via the LAN 107.

It should be noted that, as a difference to the authentication procedure described in conjunction with Fig. 1 and Fig. 4, the device 115 does not have to set up a new connection to the Internet when transmitting the E-mail comprising the code message, since an Internet connection already has been set up when transmitting the first E-mail in step 400. However, it is of course possible to let the device connect to the Internet a second time when it receives the code word in step 407, if this should turn out to be advantageous.

The E-mail from the device 115 to the authentication server 105 comprises at least the code message. The generation of the code message is preferably done by using a unique encryption key. In particular the code message is generated by encrypting the code word and a part of the encryption key.

In a preferred embodiment the E-mails from the device 115 to the authentication server 105 are transmitted using a SMTP or HTTP protocol, since the firewall 109 is designed to only let such data traffic through. Thus, the firewall 109 can maintain a high level of security. Preferably the SMTP protocol is used, since it only requires one socket to be used for the connection.

Thereupon, the authentication server receives the E-mail from the device, step 417. The code word is then extracted from the received E-mail, e.g. by extracting and decrypting the code message, step 419. Next, it is checked if the transmitted code word matches the received code word, step 421.

If the extracted code word does not match the transmitted code word the session ends, step 423. However, in a preferred embodiment the computer system 101 can be arranged to return a message to the device informing the user of the device that access is denied, so that the user can try to connect again if he/she wishes to do so.

If the extracted code word matches the code word originally transmitted to the device 115, the authentication server knows that it is the correct device. Thus, the client can start to make transactions in the server 104, step 425.

The device 115 can be any type of mobile communication device, which can receive and transmits data. Thus, the device can be a mobile telephone, a computer comprising a wireless modem or a combination thereof, such as a hand held computer comprising an integrated mobile telephone. In order for the device 115 to operate properly it must be loaded with a suitable program, which enables the device to communicate using the method as described above. The functions required for enabling the device to operate according to the method can of course also be implemented in hardware. Software or hardware support for the authentication method is also provided in the computer system.

The system as described in conjunction with Figs. 3 and 6 can also be used in other types of transactions than financial transactions. Thus, it is possible to use the system in a multitude of applications where a quick but secure authentication of a device is required. Examples on such applications are bookmaking transactions, ticket booking transactions, buying and selling of goods etc.

Finally, in Appendix 1 - 12 a computer program listing illustrating a software implementation of different procedure steps is shown. Thus, in Appendices 1 - 6 a program listing in C++ language of some procedure steps in the computer system is shown, for example "lumpio" in appendix 5-6 deals with encoding and decoding. In Appendices 7 - 12 a listing of some procedure steps in the client or device is shown, for example: "pushnot" in appendices 7-8 initiates a connection upon receipt of an SMS using the function "DoConnect"; "client" in appendices 11-12 use the functions "lumpIO_fputs" and "lumpIO_fgets" for decryption and encryption, and the function "send_file" to send a mail.

By using the method and the system as described herein, several different types of authenticated data transmission can be obtained. For example, automated forwarding of data, such as electronic mail (E-mail), facilitated login on a secure network, such as a bank and many more applications where an authentication is required or desired.

/*APPENDIX 1: pat_smtp.h*/

```
#ifndef lint
static char *rcsid_notifier_h = "$Header$";
#endif

#ifdef UNIX
# define DEFAULT_LOG_FILE "/home/lmjm/pat_smtp/log"
# define DEFAULT_CONFIG_FILE "/home/janko/pat_smtp/pat_smtp.conf"
# define DEFAULT_ACCOUNTS_FILE "/home/janko/pat_smtp/accounts"
# define DEFAULT_QUEUE_DIR "/home/janko/pat_smtp/queue"
# define DEFAULT_SERVER_VERSION "PAT Server 1.0beta"
# define MAX_ACCOUNTS 1000
#endif
#ifdef WIN32
# define DEFAULT_LOG_FILE "c:/pat/logfile.txt"
# define DEFAULT_CONFIG_FILE "c:/pat/pat_smtp.conf"
# define DEFAULT_ACCOUNTS_FILE "c:/pat/accounts.txt"
# define DEFAULT_QUEUE_DIR "c:/pat/queue"
# define DEFAULT_SERVER_VERSION "PAT Server 1.0beta"
# define MAX_ACCOUNTS 1000
#endif
#define CLIENT_DOM "ewi"

#define BIG_BUF 512

#define bool char
#define true 1
#define false 0

extern bool smtp_listener;
extern bool daemon;
extern int verbose;

extern char *config_file;
extern char *queue_dir;
extern char *server_version;

extern char *program_name;
extern int program_pid;

void read_config_file();
```

```
/*APPENDIX 2: pat_smtp.cpp */

/* SMTP listener.
 * Can run either as a permanently running process, under inetd (on UNIX) or just
 * from the command line.
 *
 * This SMTP server expects to get either:
 * [[ Step (a) is NOT needed - messages will get from
 *    the EVO server into the right place behind my back
 * a) "real" email from the Internet which must be sent
 *    on to a PAT client (on EPOC) - in which case it should queue
 *    the email to be sent AND queue an SMS to be sent to the user
 *              which will contain the token used to authenticate them
 * ]]
 *
 * b) "PAT" email from a PAT client (on EPOC) - this begins with
 *    an initial "fake" SMTP incoming message of just:
 *              HELO
 *     MAIL FROM:<epocclient_versionnumber@ewi>
 *              RCPT TO:<patserver_versionnumber@ewi>
 *              RCPT TO:<token_verification@ewi>
 *              TURN
 *             This is used to authenticate the PAT client.
 *             If this is successful then we send the pending emails and
 *             a 'TURN' so that the client can send in any pending emails.
 *             Once the client has sent any emails it will end with a QUIT.
 *
* $Log$
 */

#ifndef lint
static char *rcsid = "$Header$";
#endif

#include <stdio.h>
#include <sys/types.h>

#ifdef UNIX
# include <unistd.h>
# include <dirent.h>
# include <pwd.h>
# include <sys/time.h>
# include <sys/socket.h>
# include <netinet/in.h>
# include <arpa/inet.h>
# include <netdb.h>
# include <signal.h>
# include <sys/stat.h>
/* For getopt() */
extern char *optarg;
extern int optind;
#endif

#ifdef WIN32
```

```
# include <stdlib.h>
# include <io.h>
# include <time.h>
# include <process.h>
# include <direct.h>
# include <string.h>
# include <sys/stat.h>
# include <winsock2.h>
# include "win_support.h"
#endif

#include <string.h>
#include <ctype.h>

#include "pat_smtp.h"
#include "netio.h"
#include "keylib/keylib.h"
#include "log.h"
#include "base64.h"
#include "cryptsup.h"
#include "lumpIO.h"
#include "hex.h"

char *program_name = "pat_smtp";
int program_pid;

bool smtp_listener = true;
bool daemon = true;
int verbose = 2;

u_short smtp_port = 25;
int max_listen_backlog = 5;
int pulse_time = 5; /* break out of the accept loop every 5 seconds */
int keep_looping = 1;
int smtp_timeout = 120; /* how long to wait for input */

int skip_authentication = 1;

#define MAX_DB_FIELDLEN 100
char user[ MAX_DB_FIELDLEN ],
   password[ MAX_DB_FIELDLEN ],
        evokey[ MAX_DB_FIELDLEN ],
        randnum[ MAX_DB_FIELDLEN ],
        dbtime[ MAX_DB_FIELDLEN ];


#define MAX_VERSION 100
char client_version[ MAX_VERSION ];

bool authenticated = false;

void main( int argc, char **argv );
void parse_args( int argc, char **argv );
void become_daemon();
```

```
#ifdef UNIX
void process_under_inetd();
char *get_my_login();
#endif
void process_smtp( Sock io );
bool process_state( int state, NET n, char *buf, char *addr );
void process_queue_for_user( char *user, NET n );
bool process_queue_file( char *qfilename, NET n );
bool filename_matches( char *filename, char *user );
void to_upper( char *str );
int mail_rcpt_addr( char *buf, char *addr );
#define TMP_IN 0
#define TMP_OUT 1
FILE *create_tmp( int where, char *tmp, char *user );
int read_smtp_reply( NET n, char *buf, int bufsiz );
void init_winsock();
void rename_in_tempfile( char *tmp_name );


/* Does the SMTP reply mean OK */
#define REPLY_OK(r) ((200 <= (r)) && ((r) <= 299))

/* States that the server is in */
#define ST_INIT 0
#define ST_HELO 1
#define ST_MAIL_CLIENT_VERS 2
#define ST_RCPT_SERVER_VERS 3
#define ST_RCPT_TOKEN 4
#define ST_MAIL_OUT 5
#define ST_RCPT_OUT 6
#define ST_DATA_OUT 7
#define ST_MAIL_IN 8
#define ST_RCPT_IN 9
#define ST_DATA_IN 10


void main( int argc, char **argv )
{
#ifdef WIN32
        extern int _fmode;
        _fmode = _O_BINARY;
#endif
        */

        if( verbose ){
                netio_show( verbose );
                log_to_tty = 1;
        }

  program_pid = getpid();

  parse_args( argc, argv );

#ifdef UNIX
```

```
    if( ! smtp_listener ){
      to = get_my_login();
      (void)queue_request();
      exit( 0 );
    }

    if( daemon ){
      become_daemon();
      exit( 0 );
    }

    process_under_inetd();
#endif
#ifdef WIN32
    become_daemon();
#endif
}

void parse_args( int argc, char **argv )
{
    int c;

        char **nargv;

#ifdef UNIX
    while( (c = getopt( argc, argv, "avl:f:di:" )) != -1 ){
#endif /* UNIX */
#ifdef WIN32
    while( --argc > 0 ){
                char *arg = *++argv;
                char *optarg;
                c = *arg;
                if( c == '-' )
                        c = arg[ 1 ];
                else {
                        /* Not an argument - make nargv point at all the rest */
                        nargv = argv;
                        return;
                }
                switch( c ){
                case 'l':
                        optarg = *argv++;
                }
#endif /* WIN32 */

      switch( c ){
      case 'v':
        verbose++;
        break;
                case 'a':
                        skip_authentication = false; /* Must authenticate user */
                        break;
      case 'l':
        log_file = optarg;
```

```
            if( *log_file != '/' ){
               fprintf( stderr, "logfile name must begin with /\n" );
               exit( 1 );
            }
            break;
         case 'f':
            config_file = optarg;
            if( *config_file != '/' ){
               fprintf( stderr, "config filename must begin with /\n" );
               exit( 1 );
            }
            break;
#ifdef UNIX
         case 'd':
            /* Run as a stand-alone daemon */
            smtp_listener = true;
            daemon = true;
            break;
         case 'i':
            /* Run under inetd */
            smtp_listener = true;
            daemon = false;
            break;
#endif
         default:
            fprintf( stderr, "Usage: %s v [-l logfile] [-f configfile] [-d|-i|-c From]\n",
program_name );
            fprintf( stderr, "    -v verbose mode\n" );
            fprintf( stderr, "    -l log to this file\n" );
            fprintf( stderr, "    -f use this configuration file\n" );
#ifdef UNIX
            fprintf( stderr, "    -d run as stand-alone SMTP daemon\n" );
            fprintf( stderr, "    -i run under inetd\n" );
#endif
            fprintf( stderr, "    -a authenticated\n" );
            exit( 0 );
         }
      }

            if( verbose )
                  log_to_tty = 1;

}

void become_daemon()
{
   Sock sock, msgsock;
   struct sockaddr_in server;
   fd_set ready;
   struct timeval to;
   int on;

   read_config_file();
```

```
   logit( INFO1, "smtp daemon starting" );

#ifdef WIN32
        init_winsock();
#endif

   if( (sock = socket( AF_INET, SOCK_STREAM, 0 )) < 0 )
      logit( FATAL, "Cannot create stream socket" );

   server.sin_family = AF_INET;
   server.sin_addr.s_addr = htonl( INADDR_ANY );
   server.sin_port = htons( smtp_port );

   on = 1;
   if( setsockopt( sock, SOL_SOCKET, SO_REUSEADDR, (char *)(&on), sizeof( on ) ) < 0 )
      logit( WARNING, "Cannot turn on SO_REUSEADDR" );

   if( bind( sock, (struct sockaddr *)&server, sizeof( server ) ) < 0 )
      logit( FATAL, "Cannot bind to SMTP port %d", smtp_port );

   listen( sock, max_listen_backlog );

   while( keep_looping ){
      logit( INFO, "loop waiting for SMTP connections" );

      FD_ZERO( &ready );
      FD_SET( sock, &ready );
      to.tv_sec = pulse_time;
      to.tv_usec = 0;
      if( select( sock + 1, &ready, 0, 0, &to ) < 0 ){
         logit( WARNING, "select error" );
         continue;
      }
      if( FD_ISSET( sock, &ready ) ){
         if( (msgsock = accept( sock, (struct sockaddr *)0, (int *)0 )) < 0 )
            logit( WARNING, "failed to accept" );
         process_smtp( msgsock );
      }
   }
}


void process_smtp( Sock io )
{
        NET n;
        int state = ST_INIT;
  char buf[ BIG_BUF ];
  char tmp1[ BIG_BUF ];
  char tmp2[ BIG_BUF ];
  char rcpt_to[ BIG_BUF ];
  char mail_from[ BIG_BUF ];
  char tmp_name[ BIG_BUF ];
        FILE *tmp_f;
```

```
struct sockaddr_in remote_addr;
int addrlen = sizeof( remote_addr );
struct hostent *hp;
char remote_host[ BIG_BUF ];

logit( INFO, "process_smtp" );

/* Figure out who is calling me */
if( getpeername( io, (struct sockaddr *)&remote_addr, &addrlen ) < 0 ){
   logit( WARNING, "Cannot find remote host details" );
   strcpy( remote_host, "unknown" );
}
else {
   hp = gethostbyaddr( (char *)&(remote_addr.sin_addr),
                              sizeof( struct in_addr ), AF_INET );
   if( hp )
      (void)strncpy( remote_host, hp->h_name, sizeof( remote_host ) );
   else
      (void)strncpy( remote_host, inet_ntoa( remote_addr.sin_addr ),
                           sizeof( remote_host ) );
}

logit( INFO1, "process_smtp from %s", &remote_host[ 0 ] );

if( (n = nopen( io )) < 0 )
   logit( FATAL, "cannot use connection for smtp traffic" );


/* Talk SMTP with the remote client */

nprintf( n, "220 ME PAT_SMTP\r\n" );

while( true ){
   bool rcpt_to_cmd, mail_from_cmd;
   rcpt_to_cmd = mail_from_cmd = false;

   if( timed_read_line( n, buf, sizeof( buf ), smtp_timeout ) < 0 )
      break;

   tmp1[ 0 ] = tmp2[ 0 ] = '\0';
   sscanf( buf, "%[^ \r\n]%*[ \t]%[^\n\r]", tmp1, tmp2    );

   to_upper( tmp1 );
   logit( INFO, "cmd: %s", tmp1 );
   if( strcmp( tmp1, "HELP" ) == 0 ){
      nprintf( n, "214-Commands supported:\r\n" );
      nprintf( n, "213 HELO MAIL RCPT DATA TURN QUIT HELP\r\n" );
      continue;
   }
   else if( strcmp( tmp1, "HELO" ) == 0 ){
                        state = ST_HELO;
                        if( ! process_state( state, n, buf, "" ) )
                              break;
      continue;
```

```
    }
    else if( strcmp( tmp1, "MAIL" ) == 0 ){
      if( ! mail_rcpt_addr( buf, mail_from ) ){
        nprintf( n, "500 Bad MAIL command\r\n" );
        continue;
      }
                    if( state == ST_HELO )
                        state = ST_MAIL_CLIENT_VERS;
                    else if( state == ST_MAIL_IN ){
                        state = ST_MAIL_IN; /* A tad redundant! */
                        if( (tmp_f = create_tmp( TMP_IN, tmp_name, user )) ==
NULL ){
                          nprintf( n, "500 Cannot create tmp files\r\n" );
                            break;
                          }
                    }
                    else {
                        nprintf( n, "501 Unexpected MAIL command\r\n" );
                        continue;
                    }
                    if( ! process_state( state, n, buf, mail_from ) )
                        break;
                    continue;
    }
    else if( strcmp( tmp1, "RCPT" ) == 0 ){
      if( ! mail_rcpt_addr( buf, rcpt_to ) ){
        nprintf( n, "500 Bad RCPT command\r\n" );
        continue;
      }
                    if( state == ST_MAIL_CLIENT_VERS )
                        state = ST_RCPT_SERVER_VERS;
                    else if( state == ST_RCPT_SERVER_VERS )
                        state = ST_RCPT_TOKEN;
                    else if( state == ST_MAIL_IN ){
                        state = ST_RCPT_IN;
                    }
                    else {
                        nprintf( n, "501 Unexpected RCPT command\r\n" );
                        continue;
                    }
                    if( ! process_state( state, n, buf, rcpt_to ) )
                        break;
                    continue;
    }
    else if( strcmp( tmp1, "DATA" ) == 0 ){
      bool bad_end = false;
                    char *buf_save;

                    state = ST_MAIL_IN;   /* Await more mail after this */

                    struct lumpIO *f = lumpIO_init( key, iv, tmp_f );

      nprintf( n, "354 Enter message, ending with \".\" on a line by itself\r\n" );
      while( 1 ){
```

```
        if( timed_read_line( n, buf, sizeof( buf ), smtp_timeout ) < 0 ){
          bad_end = true;
          break;
        }
        if( strcmp( buf, ".\r\n" ) == 0 )
          break;
                                /* If the line is just '..' skip the first '.' */
                                if( strcmp( buf, "..\r\n" ) == 0 )
                                        buf_save = &buf[ 1 ];
                                else
                                        buf_save = buf;
                                lumpIO_fputs( buf_save, f );
      }
      if( bad_end )
        break;
                        fclose( tmp_f );

                        rename_in_tempfile( tmp_name );

      nprintf( n, "250 OK\r\n" );
      continue;
    }
    else if( strcmp( tmp1, "TURN" ) == 0 ){
                        if( state == ST_RCPT_TOKEN ){
                                state = ST_MAIL_OUT;
                                nprintf( n, "250 turning (sending mail to client)\r\n" );
                                /* Fall thru to the end of the main if statement */;
                        }
                        else {
                                nprintf( n, "501 Unexpected TURN command\r\n" );
                                continue;
                        }
    }
    else if( strcmp( tmp1, "QUIT" ) == 0 ){
      nprintf( n, "221 closing\r\n" );
      break;
    }
    else {
      nprintf( n, "500 Command unrecognized\r\n" );
      continue;
    }
              if( state == ST_MAIL_OUT ){
                        int reply;

                        process_queue_for_user( user, n );
                        state = ST_MAIL_IN;
                        nprintf( n, "TURN\r\n" );
      if( (reply = read_smtp_reply( n, buf, sizeof( buf ) )) < 0 )
                                break;
                        if( REPLY_OK( reply ) ){
                                continue;
                        }
                        /* client has no email to send to me! */
                        nprintf( n, "QUIT\r\n" );
```

```
                        break;
                }
        }
        nprintf( n, "200 So long and thanks for all the fish.\r\n" );
        nclose( n );

    logit( INFO, "end of process_smtp" );
}

bool process_state( int state, NET n, char *buf, char *addr )
{
        if( state == ST_HELO ){
                nprintf( n, "250 watcha!\r\n" );
                return true;
        }
        if( state == ST_MAIL_CLIENT_VERS ){
                strcpy( client_version, addr );

                authenticated = false;

                nprintf( n, "250 thanks for the client version: <%s>\r\n", addr );
                return true;
        }
        if( state == ST_RCPT_SERVER_VERS ){
                nprintf( n, "250 thanks for the server version: <%s>\r\n", addr );
                return true;
        }
        if( state == ST_RCPT_TOKEN ){
                char *at, *dom, *dot;
                char *lwidx64, *resp64;
                char idxbuf[ 100 ]; /* should be 6 bytes - ignore first 2 */
                int idxlen;
                int idx; /* The decoded index value */
                char keyidx[ 10 ]; /* string version of idx */
                int handle;
                unsigned long keyindex;
                char EvoKey_tmp[ 100 ];
                char RandomToken_tmp[ 100 ];

                if( skip_authentication ){
                        authenticated = true;
                        strcpy( user, addr );
                        nprintf( n, "250 NON-authenticated user: %s\r\n", user );

                        return true;
                }
                /* addr is: lwidx.resp64@ewi
                 * lwidx is base64(index)
                 * resp64 is base64(encrypt(hash(random)))
                 */

                authenticated = false;
                strcpy( user, "--Not Yet Set--" );
```

```
        at = strchr( addr, '@' );
        if( ! at ){
                nprintf( n, "550 bad token/address: no @\r\n" );
                return false;
        }

        dom = at + 1;
        if( strcmp( dom, CLIENT_DOM ) != 0 ){
                nprintf( n, "550 bad client domain: %s\r\n", dom );
                return false;
        }

        dot = strchr( addr, '.' );
        if( ! dot ){
                nprintf( n, "550 bad token/address: no .\r\n" );
                return false;
        }

        lwidx64 = addr;
        *dot = '\0';

        resp64 = dot + 1;
        *at = '\0';

        logit( INFO1, "got: lwidx64 %s, resp64 %s",
                lwidx64, resp64 );

        from64_init();
        int used;
        if( (idxlen = from64( lwidx64, strlen( lwidx64 ), (unsigned char *)idxbuf,
&used )) < 0 ){
                nprintf( n, "550 idx not base64\r\n" );
                return false;
        }

        if( idxlen != 6 ){
                nprintf( n, "550 idxlen should be 6 is %d for %s\r\n", idxlen, lwidx64
);
                return false;
        }

        memcpy( &idx, &idxbuf[ 2 ], 4 );
        sprintf( keyidx, "%d", idx );

        if( kbd_open_db() != OK ){
                nprintf( n, "550 Cannot access db\r\n" );
                return false;
        }

        handle = keyindex = 0;  /* Currently unused */

        /*      NOTE: In the database:
         *
         * RandomToken IS IN BASE64
```

```
                * EvoKEY IS IN HEX
                */

                if( kbd_get_key( handle,
                                                keyindex,
                                                keyidx,
                                                user,
                                                password,
                                                EvoKey_tmp,
                                                RandomToken_tmp,
                                                dbtime ) != OK ){
                        if( kbd_close_db() != OK )
                                nprintf( n, "550-Failure to close db\r\n" );
                        nprintf( n, "550 Cannot find %d in db\r\n", idx );
                        return false;
                }

                if( kbd_close_db() != OK ){
                        nprintf( n, "550 Failure to close db\r\n" );
                        return false;
                }

                logit( INFO1, "db: user %s, pass %s, evokey %s, random token %s",
                        user, password, EvoKey_tmp, RandomToken_tmp );

                fromhex( EvoKey_tmp, EvoKey );

                from64_init();
                int RandomToken_len; /* TODO: Check this gets the right value */
                if( (RandomToken_len = from64( RandomToken_tmp, strlen(
RandomToken_tmp ), (unsigned char *)RandomToken, &used )) < 0 ){
                        nprintf( n, "550 random token not base64\r\n" );
                        return false;
                }

                /* here we convert received base64(E(hash(rnd))) to 8-bit form */
                from64_init();
                if( (HashToken_str_len = from64( resp64, strlen( resp64 ), HashToken_str,
&used )) < 0 ){
                        nprintf( n, "550 response part not base64\r\n" );
                        return false;
                }

                /* here we generate the DES key */
                GenerateKey( key, 24);

                /* here we generate the DES initialisation vector */
                GenerateKey( iv, 8);

                /* now we decrypt the received 8-bit form */
                ctx_3des ctx_dec;
                unsigned char plaintext[ 256 ];
                InitKey( &ctx_dec, key, iv );
                Decrypt( &ctx_dec, plaintext, HashToken_str, 16 );
```

```
                /* now we hash-up the database retrieved rnd number */
                Hash md5( SSH_HASH_MD5 );
                md5.Add( RandomToken, 16 );
                memcpy( db_hash_rnd_str, md5.Digest(), 16 );

                /* compare hash of the db entry with the decrypted EPOC hashed rnd token*/
                int     result = memcmp( db_hash_rnd_str, plaintext, 16 );

                logit( INFO1, "db_has_rnd_str %02x%02x..., plaintext %02x%02x...",
                        db_hash_rnd_str[0], db_hash_rnd_str[1],
                        plaintext[0], plaintext[1] );

                if( result != 0 ){
                        nprintf( n, "550 Authentication FAILED! <%s> (idx %d)\r\n", user,
idx );

                        return false;
                }

                nprintf( n, "250 authenticated user: <%s> (idx %d)\r\n", user, idx );

                return true;
        }
        if( state == ST_MAIL_IN ){
                nprintf( n, "250 MAIL TO for inet: <%s>\r\n", addr );
                return true;
        }
        if( state == ST_RCPT_IN ){
                nprintf( n, "250 RCPT TO for inet: <%s>\r\n", addr );
                return true;
        }

        return true;
}

void process_queue_for_user( char *user, NET n )
{
        bool found, failed;
        char here[ BIG_BUF ];
        char qdir[ BIG_BUF ];

        logit( INFO, "process_queue for %s", user );

        if( getcwd( here, BIG_BUF ) == NULL )
                logit( FATAL, "Cannot find my current directory" );

        sprintf( qdir, "%s/out", queue_dir );
        if( chdir( qdir ) < 0 )
                logit( FATAL, "Cannot chdir to %s", qdir );

        do {
#ifdef UNIX
                DIR *d;
```

```
                struct dirent *dfile;

                found = false;
                failed = false;

                if( (d = opendir( "." )) == NULL )
                        logit( FATAL, "cannot open queue directory: %s", qdir );


                while( (dfile = readdir( d )) != NULL ){
                        char *filename = dfile->d_name;
                        if( filename_matches( filename, user ) ){
                                continue;
                        logit( INFO, "Found j file: %s", filename );
                        if( ! process_queue_file( filename, n ) ){
                                failed = true;
                                break;
                        }
                        found = true;
                }
                closedir( d );
#endif
#ifdef WIN32
                struct _finddata_t j_file;
                long hFile;

                found = false;
                failed = false;

                /* Find first j* file in current directory */
                if( (hFile = _findfirst( "j*.*", &j_file )) != -1L ){
                        if( filename_matches( j_file.name, user ) ){
                                logit( INFO, "Found j file: %s", j_file.name );
                                if( ! process_queue_file( j_file.name, n ) ){
                                        failed = true;
                                        break;
                                }
                                found = true;
                        }

        /* Find the rest of the j* files */
        while( _findnext( hFile, &j_file ) == 0 && ! failed ){
                                if( filename_matches( j_file.name, user ) ){
                                        logit( INFO, "Found j file: %s", j_file.name );
                                        if( ! process_queue_file( j_file.name, n ) ){
                                                failed = true;
                                                break;
                                        }
                                        found = true;
                                }
        }
                }
                _findclose( hFile );
#endif
```

```
        } while( found && ! failed );

        if( chdir( here ) < 0 )
                logit( FATAL, "Cannot chdir to %s", here );

        logit( INFO, "findished processing queue" );

}

/* j + user + '.' + NNN */
bool filename_matches( char *filename, char *user )
{
        char tmp[ 100 ];
        char *dot;

        strcpy( tmp, &filename[ 1 ] );
        dot = strchr( tmp, '.' );
        if( ! dot )
                return false;

        *dot = '\0';

        return strcmp( tmp, user ) == 0;
}

bool process_queue_file( char *qfilename, NET n )
{
        FILE *qf;
        char tmp[ BIG_BUF ];
        int ret;
        bool eof_ok = false;

        logit( INFO, "process_queue_file: %s", qfilename );

        logit( INFO, "Send MAIL FROM:<%s>", server_version );
        nprintf( n, "MAIL FROM:<%s>\r\n", server_version );
        ret = read_smtp_reply( n, tmp, sizeof( tmp ) );
        if( ! REPLY_OK( ret ) ){
                logit( WARNING, "bad reply to MAIL FROM of %d", ret );
                return false;
        }

        logit( INFO, "Send RCPT TO:<%s>", client_version );
        nprintf( n, "RCPT TO:<%s>\r\n", client_version );
        ret = read_smtp_reply( n, tmp, sizeof( tmp ) );
        if( ! REPLY_OK( ret ) ){
                logit( WARNING, "bad reply to RCPT TO of %d", ret );
                return false;
        }

        logit( INFO, "Send data from file: %s", qfilename );
        if( (qf = fopen( qfilename, "r")) == NULL )
                logit( FATAL, "Cannot open queue file: %s", qfilename );
```

```
        logit( INFO, "DATA" );
        nprintf( n, "DATA\r\n" );
        logit( INFO, "Read reply to DATA" );
        ret = read_smtp_reply( n, tmp, sizeof( tmp ) );
        if( ! (300 <= ret && ret <= 399) ){
                logit( WARNING, "bad reply to DATA of %d", ret );
                return false;
        }

        struct lumpIO *f = lumpIO_init( key, iv, qf );

        while( lumpIO_fgets( (unsigned char *)&tmp[ 0 ], BIG_BUF, f ) != NULL ){
#ifdef UNIX
                char *nl = strchr( tmp, '\n' );
                if( nl )
                        strcpy( nl, "\r\n" );
#endif
#ifdef WIN32
                char *nl = strchr( tmp, '\r' );
                if( nl && nl[ 1 ] != '\n' )
                        strcpy( nl, "\r\n" );
#endif
                if( feof( qf ) && nl )
                        eof_ok = true;
                if( strcmp( tmp, ".\r\n" ) == 0 )
                        strcpy( tmp, "..\r\n" );
                logit( INFO, "Send: %s", tmp );
                nprintf( n, "%s", tmp );
        }
        logit( INFO, "Send ." );
        if( eof_ok )
                nprintf( n, ".\r\n" );
        else
                nprintf( n, "\r\n.\r\n" );

        fclose( qf );

        logit( INFO, "Read reply to email" );
        ret = read_smtp_reply( n, tmp, sizeof( tmp ) );
        if( ! REPLY_OK( ret ) ){
                logit( WARNING, "bad reply to email of %d", ret );
                return false;
        }

        /* Once a job is complete rename it from j* to d* */
        strcpy( tmp, qfilename );
        *tmp = 'd';
        logit( INFO, "Rename old qfile from %s to %s", qfilename, tmp );
        if( rename( qfilename, tmp ) < 0 )
                logit( FATAL, "cannot rename %s to %s", qfilename, tmp );

        return true;
}
```

```
/* Check that buf is 'some cmd:<address>' and fill addr with the address.
 * Return true if everything is OK, otherwise false */
int mail_rcpt_addr( char *buf, char *addr )
{
        char *gt;
        int len;
  char *a = strchr( buf, ':' );
  if( ! a )
                return false;

  a++; /* skip the : */
  if( *a != '<' )
                return false;

  /* skip to > */
  gt = strchr( ++a, '>' );
  if( ! gt )
                return false;

        len = gt - a;
        strncpy( addr, a, len );
        addr[ len ] = '\0';

        return true;
}

#ifdef UNIX
void process_under_inetd()
{
  /* UNTESTED */
  logit( INFO1, "process_under_inetd" );
  process_smtp( 0 );
}

char *get_my_login()
{
  int uid = getuid();
  struct passwd *pw = getpwuid( uid );

  if( pw == NULL ){
    logit( FATAL, "cannot find login name for uid: %d", uid );
  }

  return pw->pw_name;
}
#endif

void to_upper( char *str )
{
  int c;
  while( (c = *str) ){
    if( islower( c ) )
      c = toupper( c );
    *str++ = c;
```

```
    }
}

FILE *create_tmp( int where, char *tmp, char *user )
{
        char *ret;

        if( where == TMP_IN )
                sprintf( tmp, "%s/in/t%s.XXXXXX", queue_dir, user );
        else
                sprintf( tmp, "%s/out/t%s.XXXXXX", queue_dir, user );

        if( (ret = mktemp( tmp )) == NULL )
                return NULL;

        return fopen( tmp, "w" );

}

int read_smtp_reply( NET n, char *buf, int bufsiz )
{
        int ret;

   if( timed_read_line( n, buf, bufsiz, smtp_timeout ) < 0 )
      return -1;

        logit( INFO, "read reply: %s", buf );

   sscanf( buf, "%d", &ret );

        logit( INFO, "read reply value: %d", ret );

        return ret;
}

#ifdef WIN32
void init_winsock()
{
        WORD wVersionRequested;
        WSADATA wsaData;
        int err;

        wVersionRequested = MAKEWORD( 2, 0 );

        err = WSAStartup( wVersionRequested, &wsaData );
        if( err != 0 )
                logit( FATAL, "Cannot find usable winsock DLL" );

        /* Confirm that the WinSock DLL supports 2.0.*/
        /* Note that if the DLL supports versions greater    */
        /* than 2.0 in addition to 2.0, it will still return */
        /* 2.0 in wVersion since that is the version we      */
        /* requested.                                        */
```

```
        if( LOBYTE( wsaData.wVersion ) != 2 ||
                HIBYTE( wsaData.wVersion ) != 0 ) {
            WSACleanup();
            logit( FATAL, "Cannot find usable winsock DLL - wrong version" );
            return;
        }

        /* The WinSock DLL is acceptable. Proceed. */
}
#endif

/* Once a job is complete rename it from
 * ../in/t<user>.<uniq> to .../in/j.<user>.<NNN>..
 */
void rename_in_tempfile( char *tmp_name )
{
        char *sl, *num;
  char buf[ BIG_BUF ];
        struct stat sb;
        int n;

        strcpy( buf, tmp_name );

        sl = strrchr( buf, '/' );
        sl[ 1 ] = 'j';

        num = strchr( buf, '.' );
        num++;

        for( n = 0; n <= 999; n++ ){
            sprintf( num, "%03d", n );


            if( stat( buf, &sb ) < 0 )
                /* Found an unused filename! */
                break;
        }
        if( n >= 999 )
            logit( FATAL, "No free filenames!" );

        logit( INFO, "Rename incoming from %s to %s", tmp_name, buf );
        if( rename( tmp_name, buf ) < 0 )
            logit( FATAL, "No free filenames!" );

}
```

/* APPENDIX 3: cryptsup.h */

```
#ifndef CRYPTSUP_H
#define CRYPTSUP_H

#include "patparts/hash.h"
#include "patparts/des.h"
#include "patparts/getput.h"
struct ctx_3des
{
        des_key_schedule ks1, ks2, ks3;
        unsigned char iv[8];
};
void InitKey(ctx_3des *ctx, unsigned char *key, unsigned char *iv);
void Encrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len);
void Decrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len);
void GenerateKey(unsigned char *key, int keylen);
int des_ede_encrypt(unsigned char *input, unsigned char *output, des_key_schedule ks1,
                                        des_key_schedule ks2, des_key_schedule ks3, int
encrypt);
void des3_cbc(unsigned char *dest,
        const unsigned char *src, int len,
                        des_key_schedule ks1, des_key_schedule ks2,
                        des_key_schedule ks3, unsigned char *iv,
                        int for_encryption);

extern unsigned char RandomToken[ 16 ];
extern unsigned char db_hash_rnd_str[ 16 ];
extern unsigned char HashToken_str[ 16 ];
extern int HashToken_str_len;
extern unsigned char EvoKey[ 8 ];
extern unsigned char key[ 24 ], iv[ 8 ];

#endif
```

```
/* APPENDIX 4: cryptsup.cpp */


#include <memory.h>

#include "log.h"
#include "hex.h"
#include "cryptsup.h"
#include "patparts/hash.h"

unsigned char RandomToken[ 16 ];
unsigned char db_hash_rnd_str[ 16 ];
unsigned char HashToken_str[ 16 ];
int HashToken_str_len;
unsigned char EvoKey[ 8 ];
unsigned char key[ 24 ], iv[ 8 ];

void GenerateKey(unsigned char *key, int keylen)
{
        Hash md5(SSH_HASH_MD5);

        unsigned char *k = key;
        int klen = keylen;
        dumphex( "GenerateKey: EvoKey ", EvoKey, sizeof( EvoKey ) );
        logit( INFO, "ch %04x, keylen %d", ch, keylen );
        dumphex( "RandomToken", RandomToken, sizeof( RandomToken ) );

        md5.Add(EvoKey, sizeof(EvoKey));
        md5.Add(RandomToken, sizeof(RandomToken));
        memcpy(key, md5.Digest(), (keylen >= 16) ? 16 : keylen);

        dumphex( "key so far", key, keylen );

        for (int len = 16; (keylen -= 16) > 0; len += 16) {
                md5.Init(SSH_HASH_MD5);
                md5.Add(EvoKey, sizeof(EvoKey));
                md5.Add(key, len);
                memcpy(key+len, md5.Digest(), (keylen >= 16) ? 16 : keylen);
        }

        dumphex( " => key: ", key, klen );
}




int des_ede_encrypt(unsigned char *input, unsigned char *output, des_key_schedule ks1,
                                        des_key_schedule ks2, des_key_schedule ks3, int
encrypt)
{
        unsigned long ll[2];

        ll[0] = GetLong_Little(input);
```

```
        ll[1] = GetLong_Little(input+4);
        des_encrypt(ll,ll,ks1,encrypt);
        des_encrypt(ll,ll,ks2,!encrypt);
        des_encrypt(ll,ll,ks3,encrypt);
        PutLong_Little(output, ll[0]);
        PutLong_Little(output+4, ll[1]);

        return(0);
}

void des3_cbc(unsigned char *dest,
        const unsigned char *src, int len,
                        des_key_schedule ks1, des_key_schedule ks2,
                        des_key_schedule ks3, unsigned char *iv,
                        int for_encryption)
{
        int n;

        if (for_encryption) {
                while (len > 0) {
                        unsigned char input[8];
                        for (n = 0; n < 8; ++n)
                                input[n] = src[n] ^ iv[n];
                        des_ede_encrypt(input, iv, ks1, ks2, ks3, 1);
                        memcpy(dest, iv, 8);
                        src += 8;
                        dest += 8;
                        len -= 8;
                }
        }
        else {
                while (len > 0) {
                        unsigned char output[8];
                        des_ede_encrypt((unsigned char *)src, output, ks1, ks2, ks3, 0);
                        for (n = 0; n < 8; ++n)
                                dest[n] = output[n] ^ iv[n];
                        memcpy(iv, src, 8);
                        src += 8;
                        dest += 8;
                        len -= 8;
                }
        }
}



void InitKey(ctx_3des *ctx, unsigned char *key, unsigned char *iv)
{
                des_set_key((des_cblock *)key, ctx->ks1);
                des_set_key((des_cblock *)(key+8), ctx->ks2);
                des_set_key((des_cblock *)(key+16), ctx->ks3);
                memcpy(ctx->iv, iv, 8);
}
```

```
void Encrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len)
{
        des3_cbc(dest, src, len, ctx->ks1, ctx->ks2, ctx->ks3, ctx->iv, 1);
}

void Decrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len)
{
        des3_cbc(dest, src, len, ctx->ks3, ctx->ks2, ctx->ks1, ctx->iv, 0);
}
```

/* APPENDIX 5: lumpio.h/

```
#ifndef lumpIO_H
#define lumpIO_H

#ifndef lint
static char *rcsid_lumpIO_h = "$Header$";
#endif

#include "cryptsup.h"

struct lumpIO
{
        FILE *f;
        unsigned char key[ 24 ], iv[ 8 ];
        ctx_3des ctx_dec;
};


struct lumpIO *lumpIO_init( unsigned char key[ 24 ], unsigned char iv[ 8 ], FILE *f );
unsigned char *lumpIO_fgets( unsigned char *buf, int buflen, struct lumpIO *lio );
int lumpIO_fputs( char *string, struct lumpIO *lio );
#endif
```

```
/* APPENDIX 6: lumpio.cpp */


#ifndef lint
static char *rcsid = "$Header$";
#endif

#include <stdio.h>
#include <memory.h>
#include <string.h>

#include "lumpIO.h"
#include "base64.h"
#include "log.h"

struct lumpIO lumpIO;

struct lumpIO *
        lumpIO_init( unsigned char key[ 24 ], unsigned char iv[ 8 ], FILE *f )
{
        memcpy( lumpIO.key, key, 24 );
        memcpy( lumpIO.iv, iv, 8 );

        /* here we generate the DES key */
        GenerateKey( lumpIO.key, 24);

        /* here we generate the DES initialisation vector */
        GenerateKey( lumpIO.iv, 8);

        InitKey( &lumpIO.ctx_dec, lumpIO.key, lumpIO.iv );

        lumpIO.f = f;

        return &lumpIO;
}

/* Read and encrypt and base64 encode data from the file.
 * The encrypt routines can only handle data that is in multiples
 * of 8 AND 6 so fread() in chunks of data, encrypt and encode it.
 * But since the last block is probably partially filled its length
 * must be transmitted - so best to transmit the length in all blocks.
 * So the data sent is length, unused, 72 bytes data = 74 bytes.
 */
#define LUMP_LENGTH 74
#define LUMP_DATA_LEN (LUMP_LENGTH - 2)
struct lump
{
        char length;
        char unused;
        unsigned char data[ LUMP_DATA_LEN ];
};

unsigned char *lumpIO_fgets( unsigned char *buf, int buflen, struct lumpIO *lio )
{
```

```
        struct lump l;
        unsigned char tmp_in[ LUMP_DATA_LEN ];

        if( buflen < LUMP_DATA_LEN )
                return NULL;

        if( sizeof( lump ) != LUMP_LENGTH )
                logit( FATAL, "sizeof( lump ) = %d  !=  LUMP_LENGTH = %d",
                        sizeof( lump ), LUMP_LENGTH );

        FILE *f = lio->f;

        if( feof( f ) || ferror( f ) )
                return NULL;

        int in = fread( &tmp_in[ 0 ], sizeof( char ), LUMP_DATA_LEN, f );
        if( in < 0 )
                return NULL;

        logit( INFO, "fread %d bytes <<%*.*s>>", in, in, in, &tmp_in[ 0 ] );

        l.length = (char)in;
        l.unused = 0;
        Encrypt( &lio->ctx_dec, &l.data[ 0 ], &tmp_in[ 0 ], LUMP_DATA_LEN );

        to64( (unsigned char *)&l, LUMP_LENGTH, buf );

        strcat( (char *)buf, "\r\n" );

        return buf;
}

/* Take a base64, encrypted string and unbase64 & unencrypt it before writing
 * it to a file.
 *  The basic line is a lump as written by lumpIO_fgets.
 */
int lumpIO_fputs( char *str, struct lumpIO *lio )
{
        struct lump l;
        unsigned char tmp_out[ LUMP_DATA_LEN ];

        FILE *f = lio->f;

        if( feof( f ) || ferror( f ) )
                return EOF;

        if( sizeof( lump ) != LUMP_LENGTH )
                logit( FATAL, "sizeof( lump ) = %d  !=  LUMP_LENGTH = %d",
                        sizeof( lump ), LUMP_LENGTH );

        from64_init();
        int used, from;
        if( (from = from64( str, strlen( str ), &tmp_out[ 0 ], &used )) < 0 )
                return -1;
```

```
  l.length = tmp_out[ 0 ];
        Decrypt( &lio->ctx_dec, (unsigned char *)&l.data[ 0 ], &tmp_out[ 2 ],
LUMP_DATA_LEN );

        int ret = fwrite( &l.data[ 0 ], sizeof( char ), l.length, f );
        if( ret < 0 )
                return EOF;

        return ret;
}
```

/*APPENDIX 7: pushnot.h */

```
#ifndef __FLOAT_H__
#define __FLOAT_H__

#include "sms.h"
#include "smsstat.h"
#include <s32file.h>
#include <coecntrl.h>
#include <coeccntx.h>
#include <eikappui.h>
#include <eikapp.h>
#include <eikdoc.h>
#include <eikmover.h>


#define PUSH_ID 0x1000508

#define KEEP_ON      1              // when set to one disables the Psion's automatic
switch off


// various stages of connection
enum { CONNECT0,CONNECT1,CONNECT2,CONNECT3,CONNECT4 };

class CGuiViewColumnListBoxControl;


////////////////////////////////////////////////////////////////////////////
//
// Class CFloatCloseAppUi
//
// A "Close" button that fires the AppUi exit function
//

class CFloatAppUi;

class CFloatCloseAppUi : public CCoeControl
        {
public:
        CFloatCloseAppUi(CFloatAppUi* anAppUi);

public: // from CCoeControl
        void Draw(const TRect& aRect) const;
        void HandlePointerEventL(const TPointerEvent& aPointerEvent);

private:
        CFloatAppUi*  iControllingAppUi;
        };

        enum {EFloatTitleHeight=21};
        enum {EFloatMargin=10};
        enum {EFloatWidth=210,EFloatHeight=110}; // initial size of display in pixels
```

```
        enum
                {
                EFloatWindowWidth=EFloatWidth+2*EFloatMargin,
                EFloatWindowHeight=EFloatHeight+2*EFloatMargin+EFloatTitleHeight,
                };
        enum {EFloatBorderWidth=1};

//////////////////////////////////////////////////////////////////////////
//
// class CFloatControl
//
// The control class for the CFloatWindow.
//

//////////////////////////////////////////////////////////////////////////
//
// class CFloatAppUi
//
// Float application user interface

class CNewControl;
class CNotifier;

class CFloatAppUi : public CEikAppUi
   {
public:
        CFloatAppUi();
        ~CFloatAppUi();
   void ConstructL();
        void FloatExit();
        void DoTest();
        void AddLine(char *Text);
        void AddLine(TDesC8& Text);
        TInt TextWidth(char *Text);

        CNewControl *GetControl() { return iNewControl; };

private:
        void HandleCommandL(TInt aCommand);

private: // to provide functionality
        void CmdDlgRateL();
        void CmdDlgFileSaveAsL();
        void FloatIn();
        void FloatOut();
        void SetWindowSizeL(TSize);
        TSize WindowSize();
public:
        RFs     fileserver;
        TInt ConnectionStatus;
        CSMSStats PhoneStat;
        ESMSUtil* SMSUtils;
        CPeriodic* SmsTimer;
private:
```

```
  CNewControl* iNewControl;
        TTimeIntervalMicroSeconds32 iInterval;
        RThread SMSThread;
        CGuiViewColumnListBoxControl* iColumnView;
private:
        enum {
                EFloatFactorDelta = 200,
                EWindowSizeDelta = 2
        };
  };

//////////////////////////////////////////////////////////////////////////
//
// class CFloatDocument
//

class CFloatDocument : public CEikDocument
        {
public:
        CFloatDocument(CEikApplication& aApp) : CEikDocument(aApp) { }
private: // from CApaDocument
        CEikAppUi* CreateAppUiL();
        };


//////////////////////////////////////////////////////////////////////////
//
// class CFloatApplication
//


class CFloatApplication : public CEikApplication
        {
private:
        CApaDocument* CreateDocumentL();
        TUid AppDllUid() const;
        };

const TUid KUidFloatApp={PUSH_ID};



        enum {ENewTitleHeight=21};
        enum {ENewMargin=5};
        enum {ENewWidth=210,ENewHeight=60}; // initial size of display in pixels
        enum
                {
                ENewWindowWidth=ENewWidth+2*ENewMargin,
                ENewWindowHeight=ENewHeight+2*ENewMargin+ENewTitleHeight,
                };
        enum {ENewBorderWidth=1};
```

```
////////////////////////////////////////////////////////////////////////////
//
// class CNewWindow
//

class CNewWindow : public CCoeControl
{
        enum { ETextBoxWidth=180 };

public:
        ~CNewWindow();
        CNewWindow();
        void ConstructL();
        void SetActive(TBool aActive);
        TBool Active() const;
        TSize MinimumSize();
private: // overridden
        void Draw(const TRect& aRect) const;
        void HandlePointerEventL(const TPointerEvent& aPointerEvent);
protected:
        void SizeChangedL();
private:
        TBuf<80> iText;
        TBool iActive;
        TRect iScreenRect;
        CFbsBitmap *iBitmap;
};


////////////////////////////////////////////////////////////////////////////
//
// class CNewControl
//
// The control class for the CNewWindow.
//

class CNewControl : public CCoeControl,public MCoeControlObserver
   {
public:
public: // From CCoeControl
        CNewControl(CFloatAppUi* anAppUi);
        ~CNewControl();
   void ConstructL();
        void SizeChangedL();
        virtual TInt CountComponentControls() const;
        virtual CCoeControl* ComponentControl(TInt aIndex) const;
        virtual TKeyResponse OfferKeyEventL(const TKeyEvent& aKeyEvent,TEventCode
aType);
        virtual TSize MinimumSize();
        void    HandleControlEventL(CCoeControl* aControl, TCoeEvent aEventType);
        CEikCommandButton* NewButton(const TDesC8& text);
        inline CNewWindow* GetWindow() { return iNewWindow; };
```

```
private:
        void Draw(const TRect& aRect) const;
private:
        TSize       iWindowSize;
        TRect                   iNewRect;
        CEikMover*              iMover;
        CEikCommandButton* iTestButton;
        CNewWindow*   iNewWindow;
        CFloatAppUi*  iParentAppUi;
        CFloatCloseAppUi*     iClose;
};

#define POLL_PHONE_RATE 300000 // number of microseconds between polling phone

TInt PollPhone(TAny *APtr);                    // the periodic timer callback function.



#endif
```

/* APPENDIX 8: pushnot.cpp */

```
#include <eikchlst.h>
#include <eikopbut.h>
#include <eikdlgtb.h>
#include <eikmenu.hrh>
#include <eikcmds.hrh>
#include <eikchlst.hrh>
#include <eikdialg.h>
#include <coecntrl.h>
#include <eikappui.h>
#include <eikdoc.h>
#include <eikenv.h>
#include <eikdef.h>
#include <eikapp.h>
#include <eikmover.h>
#include <eikrubtl.h>
#include <badesca.h>
#include <basched.h>
#include <barsread.h>
#include <e32hal.h>
#include <e32std.h>
#include <e32base.h>
#include <frmframe.h>
#include <eikcfdlg.h>
#include <eikon.rsg>
#include <eikcmbut.h>
#include <eiklabel.h>
#include <eikpgsel.h>
#include "pushnot.hrh"
#include "pushnot.h"
#include <pushnot.rsg>
#include "bmaps.h"
#include "euclsv.h"
#include "socket.h"
#include "noteserv.h"
#include "iobox.h"
#include "smsutil.h"
#include "crypt.h"
#include "client.h"
#include "listwin.h"

#define PAT_IP_ADDRESS     ((195*0x1000000)+(50*0x10000)+(91*0x100)+208)

void DoConnect(CFloatAppUi* ParentAppUi);
TInt ThreadFunction(TAny* anArg);
void StartThread(CFloatAppUi* ParentAppUi,RThread *thread);


void CFloatAppUi::AddLine(TDesC8& Text)
{
        char *Txt=(char *)Text.Ptr();
```

```
        Txt[Text.Length()]=0;

        AddLine(Txt);

//      iColumnView->GetListArray()->AppendL(Buffer);
//      iColumnView->SetCurrentItem(9000);
//      iColumnView->RedrawL();
}



#define IsPunct(a) ((a==' ') || (a==',') || (a=='.') || (a=='?') || (a=='!') || (a==':') || (a==';'))

TInt CFloatAppUi::TextWidth(char *Text)
{
        TPtrC8* Buffer=new TPtrC8((TUint8 *)Text);
        TInt width=iEikonEnv->Static()->NormalFont()->TextWidthInPixels(*Buffer);
        delete Buffer;
        return(width);
}

#define SCREEN_WIDTH      380

void CFloatAppUi::AddLine(char *Text)
{
        char KeepChar;
        TInt TextPtr=0,BestSoFar=0;


        TInt width=TextWidth(Text);


        if (width>SCREEN_WIDTH)
        {
                do
                {


                        // find end of string or punctuation

//                      while ((Text[TextPtr]!=0) && (!(IsPunct(Text[TextPtr]))))
//                      {
//                              TextPtr++;
//                      }

                        // exit loop if end found
                        if (Text[TextPtr]==0) break;

                        KeepChar=Text[TextPtr];
                        Text[TextPtr]=0;

                        // and find out it's width now
                        width=TextWidth(Text);
```

```
                    Text[TextPtr]=KeepChar;

                    // if it fit's on the screen keep the position
                    if (width<SCREEN_WIDTH) BestSoFar=TextPtr;
                    TextPtr++;

                } while (width<SCREEN_WIDTH);      // but keep looking until we wrap

                if (BestSoFar!=0)
                {
                        KeepChar=Text[BestSoFar];
                        Text[BestSoFar]=0;
                        TBufC8 <512> Buffer((const unsigned char *)Text);
                        iColumnView->GetListArray()->AppendL(Buffer);
                        iColumnView->RedrawL();
                        Text[BestSoFar]=KeepChar;

                        AddLine(&Text[BestSoFar]);
                        return;
                }
        }

        TBufC8 <512> Buffer((const unsigned char *)Text);
        iColumnView->GetListArray()->AppendL(Buffer);

        iColumnView->SetCurrentItem(9000);
        iColumnView->RedrawL();
}



//RSemaphore semaphore;

#pragma data_seg(".E32_UID")
__WINS_UID(0x10000079,KAppUidValue,PUSH_ID)
#pragma data_seg()


#define WANT_WINDOW


#define MAIL_W          50
#define MAIL_H          53
#define BYTES_W         50
#define BYTES_H         53
#define STAT_W          100
#define STAT_H          35
#define PHONE_W         (EFloatWidth-STAT_W-4)
#define PHONE_H         35

#define SPACER          4       // gap between jigsaw pieces and mail/units boxes
```

```
#define TILES_WIDTH         (EFloatWidth-(MAIL_W+BYTES_W+2*SPACER))

// x position of left tile when broken apart
#define X_LEFT              ((EFloatWindowWidth-TILES_WIDTH)/2)
// x position of centre tile at all times
#define X_CENTRE    ((EFloatWindowWidth-CentreBMWidth)/2)
// x psoition of right tile when broken apart
#define X_RIGHT             ((EFloatWindowWidth+TILES_WIDTH)/2-RightBMWidth)
// x position of left tile when closed
#define X_LEFT3             (X_CENTRE-LeftBMWidth+15)
// x position of left tile when closing
#define X_LEFT2             (X_LEFT+(X_LEFT3-X_LEFT)/2)
// x position of right tile when closed
#define X_RIGHT3    (X_CENTRE+69)
// x position of right tile when closing
#define X_RIGHT2            (X_RIGHT+(X_RIGHT3-X_RIGHT)/2)


#define PHONE_BAR_W             (PHONE_W-74)/2
#define BATTERY_BAR_W   (PHONE_W-74)/2

#define SMALL_PIECE         4
#define GAP                 2

#define BM_X            MAIL_W+5

#define BLOCK_HEIGHT        8
#define BLOCK_WIDTH             6
#define BLOCK_SPACE             2

#define BUTTON_TOP_MARGIN           6       // margin between title and buttons
#define BUTTON_BOTTOM_MARGIN        0       // margin between buttons and
window

#define TOTAL_MARGIN
        (BUTTON_TOP_MARGIN+BUTTON_BOTTOM_MARGIN)

#define KFloatDefaultPath _L("C:\\")


int Wait(TTimeIntervalMicroSeconds32 Delay);

int Wait(TTimeIntervalMicroSeconds32 Delay)
{
        RTimer timer;
        TRequestStatus TStat;

        timer.CreateLocal();

        timer.After(TStat,Delay);

        User::WaitForRequest (TStat);
        timer.Close();
        return(TRUE);
```

```
}


////////////////////////////////////////////////////////////////////////////
//
// Class CFloatCloseAppUi
//
// A "Close" button that fires the AppUi exit function
//

CFloatCloseAppUi::CFloatCloseAppUi(CFloatAppUi* anAppUi)
        : iControllingAppUi(anAppUi)
        {
        __DECLARE_NAME(_S("CFloatCloseAppUi"));
        SetNonFocusing();
        }

void CFloatCloseAppUi::Draw(const TRect& /*aRect*/) const
{
        CWindowGc& gc=SystemGc();

        // draws the surrounding box
        TRect rect=Rect();
        gc.DrawRect(rect);
        TPoint pos=Position();
        gc.MoveTo(TPoint(pos.iX,rect.Size().iHeight));
        gc.DrawLineBy(TPoint(rect.Size().iWidth,-rect.Size().iHeight));
        gc.MoveBy(TPoint(0,rect.Size().iHeight));
        gc.DrawLineBy(TPoint(-rect.Size().iWidth,-rect.Size().iHeight));
}

void CFloatCloseAppUi::HandlePointerEventL(const TPointerEvent& aPointerEvent)
        {
        if (aPointerEvent.iType == TPointerEvent::EButton1Down)
                iControllingAppUi->FloatExit();
        }



////////////////////////////////////////////////////////////////////////////
//
// class CFloatAppUi
//


CFloatAppUi::CFloatAppUi()
        {
//      iInterval = 20*100000;
        }
```

```
void CFloatAppUi::DoTest()
{
/*      HBufC *tmp=HBufC::New(256);

        PatClient Pat;
        CWSocket* Socket;

        *tmp=(_L("Hello"));

        Pat.timed_read_line(Socket,tmp->Des(),10);


        delete tmp;*/

//      PatClient Pat;
//      Pat.GenFCount(fileserver);
}


void CFloatAppUi::ConstructL()
  {

        User::LeaveIfError (fileserver.Connect ());


//      DoTest();

        SMSUtils=new ESMSUtil(&this->PhoneStat);

#ifndef __WINS__
        SMSUtils->OpenPort();          // open the infra-red and configure it
#endif

  BaseConstructL();


        iColumnView = new(ELeave) CGuiViewColumnListBoxControl(this);
        iColumnView->ConstructL(ClientRect());
        AddToStackL(iColumnView);
        iColumnView->ActivateL();


        iColumnView->GetListArray()->Reset();

        iColumnView->RedrawL();



  iNewControl=new(ELeave) CNewControl(this);
  iNewControl->ConstructL();
        iNewControl->ActivateL();


        AddToStackL(iNewControl);
```

```
        PhoneStat.SetSignal(25);
        PhoneStat.SetBattery(75);
        PhoneStat.SetMessages(0);
/*      if ((Notifier=StartThread(&PhoneStat,&NoteThread))==NULL)
                    CEikonEnv::Beep();*/


        // install a periodic timer giving it a pointer to our AppUi as it's parameter
        TCallBack cb(PollPhone,this);

        SmsTimer=CPeriodic::NewL(-10);
        SmsTimer->Start(POLL_PHONE_RATE,POLL_PHONE_RATE,cb);

//      StartThread(this,&SMSThread);

}

TInt PollPhone(TAny *APtr)
{
//      UserHal::ModifyLedMask(KLedMaskGreen1,KLedMaskRed1);


#if (KEEP_ON==1)
        UserHal::ResetAutoSwitchOffTimer();
#endif

        CFloatAppUi* ParentAppUi=(CFloatAppUi *)APtr;


        // run the sms state machine

        CNewWindow* Win=ParentAppUi->GetControl()->GetWindow();

#ifdef  WANT_SERIAL
        ParentAppUi->SMSUtils->DoSMSState(Win);
#else
        //      Wait(2000000);

#endif


        CSMSUtil* SMS=ParentAppUi->SMSUtils->GetSMSUtil();

        if (ParentAppUi->SMSUtils->SMS_State==SERIAL_CLEAR)
        {
                ParentAppUi->SMSUtils->SMS_State=DO_NOTHING;// reset state
machine

                if (SMS->gottoken)
                {
```

```
                    SMS->gottoken=FALSE;
                    DoConnect(ParentAppUi);
                }

        }

        return(TRUE);
}



CFloatAppUi::~CFloatAppUi()
        {
//          delete iFloatControl;
                delete iNewControl;
                delete SmsTimer;
            delete iColumnView;

#ifndef __WINS__
                SMSUtils->ClosePort();
#endif
                delete SMSUtils;

                SMSThread.Terminate(0);
//              Notifier->Close();
}

void CFloatAppUi::HandleCommandL(TInt aCommand)
        {
        switch (aCommand)
                {

  case EEikCmdExit:
                FloatExit();


                }
        }


void CFloatAppUi::SetWindowSizeL(TSize aControlSize)
        {
        iNewControl->SetExtentL(iNewControl->PositionRelativeToScreen(),
aControlSize);
        }

TSize CFloatAppUi::WindowSize()
{
        return iNewControl->Size();
}


void CFloatAppUi::FloatExit()
```

```
{
        iCoeEnv->Flush(200000);
   CBaActiveScheduler::Exit();
}


////////////////////////////////////////////////////////////////////////////
//
// class CFloatDocument
//

CEikAppUi* CFloatDocument::CreateAppUiL()
        {
   return(new(ELeave) CFloatAppUi);
        }

////////////////////////////////////////////////////////////////////////////
//
// class CFloatApplication
//

TUid CFloatApplication::AppDllUid() const
        {
        return(KUidFloatApp);
        }

CApaDocument* CFloatApplication::CreateDocumentL()
        {
        return(new(ELeave) CFloatDocument(*this));
        }



////////////////////////////////////////////////////////////////////////////
//
// EXPORTed functions
//

EXPORT_C CApaApplication* NewApplication()
        {
        return(new CFloatApplication);
        }

GLDEF_C TInt E32Dll(TDllReason)
        {
        return(KErrNone);
        }


////////////////////////////////////////////////////////////////////////////
//
// class CNewWindow
```

```
//
// The window which displays the Float-in-area. Includes all the code for getting
// the pointer events, updating the Float-in-area indicator, and the grabbing and
// displaying of the bitmap.
//


CNewWindow::CNewWindow()
//
// Constructor
//
{
        __DECLARE_NAME(_S("CNewWindow"));
//      SetNonFocusing();

}

CNewWindow::~CNewWindow()
//
// Destructor
//
{
        delete iBitmap;
}

void CNewWindow::ConstructL()
//
// load bitmap for displaying Floated area
//
{
        iText=_L("Notifier State");

        TBufC<40> BitmapFile(_L("\\evo\\allbms.mbm")); // !! EIKON service to do this in
a soft-coded manner is forthcoming
        // set up name for bitmap sharing
        TBool shareIfLoaded(ETrue);
        TParse    mbfn;
        TBufC<2>  c_drive(_L("C:"));
        mbfn.Set(BitmapFile,&c_drive,NULL);

        iBitmap =new (ELeave) CFbsBitmap();
        iBitmap->Load(mbfn.FullName(),EMbmBmapsBat,shareIfLoaded);

}


void CNewWindow::SetActive(TBool aActive)
//
// Activates the pointer grab flag
//
{
        iActive = aActive;
}
```

```
inline TBool CNewWindow::Active() const
//
// Returns the active flag
//
        {
        return iActive;
        }


TSize CNewWindow::MinimumSize()
//
// Minimum size of component
//
   {
        // Minimum size is the 1-pixel outline of the window.
        return TSize(1,1);
   }

void CNewWindow::SizeChangedL()
        {
        }

/*void CNewWindow::UpdateNow(TPtrC NewText)
{
        iText=NewText;
        DrawNow();
//      ControlEnv()->WsSession().Flush();
}*/

void CNewWindow::HandlePointerEventL(const TPointerEvent& aPointerEvent)
{
//      if (!Active())
//              return;

        if (aPointerEvent.iType==TPointerEvent::EButton1Up)
                {
                }
        else if (aPointerEvent.iType==TPointerEvent::EButton1Down)
                {
                        SetFocus(ETrue);
                }
        else if (aPointerEvent.iType==TPointerEvent::EDrag)
                {
//              SetFloatCentre(aPointerEvent.iParentPosition);
                }
   }

void CNewWindow::Draw(const TRect& /* aRect */) const
//
// Draws the window
//
{
        CWindowGc& gc=SystemGc();
```

```
        // draws the surrounding box
        TRect rect=Rect();
//      rect.Shrink(ENewMargin,ENewMargin);
        gc.DrawRect(rect);

        TInt LeftX=rect.iTl.iX+ENewMargin;

        const CFont *Font=iEikonEnv->LegendFont();

   gc.SetPenColor(KRgbBlack);
        // set pen colour
        gc.SetBrushStyle(CGraphicsContext::ENullBrush);
        gc.UseFont(Font);
                        // and font


                                        // the text takes up.

        TRect BoxRect(TPoint(LeftX+2,(rect.iBr.iY-Font->HeightInPixels()-
8)),TSize(ETextBoxWidth,Font->HeightInPixels()+6));

   gc.SetBrushColor(KRgbWhite);
                // set pen colour
        gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
        gc.DrawRect(BoxRect);

        gc.DrawText(iText,TPoint(LeftX+5,rect.iBr.iY-Font->DescentInPixels()-5));    //
then draw it

        TPoint pos(LeftX+2,rect.iTl.iY+2);
//      gc.BitBlt(pos,iBitmap);


}


////////////////////////////////////////////////////////////////////////////
//
// class CNewControl
//
// The control class for the CNewWindow.
//

CNewControl::CNewControl(CFloatAppUi* anAppUi)
        : iParentAppUi(anAppUi)
//
// Constructor
//
        {
        __DECLARE_NAME(_S("CNewControl"));
        }


CNewControl::~CNewControl()
```

```
//
// Destructor
//
        {
        delete iMover;
        delete iClose;
        delete iNewWindow;
        delete iTestButton;
        }

CEikCommandButton* CNewControl::NewButton(const TDesC8& text)
{
  CEikCommandButton* iButton=new(ELeave) CEikCommandButton;
  iButton->SetContainerWindowL(*this);
        iButton->SetObserver((class MCoeControlObserver *)this);
  iButton->SetTextL(text);
        iButton->Label()->iAlignment = EHCenterVTop;

        return(iButton);
}

void CNewControl::ConstructL()
//
// The actual constructor for CNewControl
//
  {
        // parent window

//      iCoeEnv=aCoeEnv;

        CreateWindowL();
        EnableDragEvents();
        Window().SetPointerGrab(ETrue);
        Window().ClaimPointerGrab();
  Window().EnableBackup();
        iEikonEnv->AddWindowShadow(this);

        // moving bar component
        iMover=new(ELeave) CEikMover;
        iMover->SetContainerWindowL(*this);
                {
                TBuf<32> title;
                iEikonEnv->ReadResource(title, R_NEW_TITLE);
                iMover->SetTextL(title);
                }
        iMover->SetActive(ETrue);


 iTestButton=NewButton(_L("Test"));


      TSize dialbuttonsize=iTestButton->MinimumSize();
```

```
        iNewRect.SetRect(       ENewMargin,
                                                ENewMargin+ENewTitleHeight,
                                                ENewMargin+ENewWidth,

        ENewMargin+ENewTitleHeight+ENewHeight);


#ifdef  WANT_WINDOW
        // New-in-area component
        iNewWindow = new (ELeave) CNewWindow();
        iNewWindow->ConstructL();
        iNewWindow->SetContainerWindowL(*this);
        iNewWindow->SetRectL(iNewRect);
        iNewWindow->SetActive(ETrue);
#endif

        // nuke component
        iClose = new (ELeave) CFloatCloseAppUi(iParentAppUi);
        iClose->SetContainerWindowL(*this);
        iClose->SetRectL(TRect(         ENewWindowWidth - ENewTitleHeight, 0,

        ENewWindowWidth,ENewTitleHeight));

        TSize containersize(ENewWindowWidth,ENewWindowHeight);

        containersize.iHeight += (iTestButton-
>MinimumSize().iHeight+TOTAL_MARGIN);

        SetExtentL(TPoint(0,0),containersize);


        TPoint buttonposition;
        buttonposition.iX = (containersize.iWidth - (dialbuttonsize.iWidth+ENewMargin)-2);
        buttonposition.iY = (containersize.iHeight-(dialbuttonsize.iHeight))/2;
//(containersize.iHeight - (dialbuttonsize.iHeight+BUTTON_TOP_MARGIN));

        iTestButton->SetExtentL(buttonposition,dialbuttonsize);// set extent for button.

        SetPosition(TPoint(420,5));

        }


void CNewControl::SizeChangedL()
//
// Resets the size of the mover for a changed window size
//
{
        TRect rect=Rect();
        rect.Shrink(1,1);
        rect.iBr.iY = rect.iTl.iY+ENewTitleHeight;
        rect.iBr.iX -= ENewTitleHeight;
        iMover->SetRectL(rect);
        rect.iTl.iX = rect.iBr.iX;
```

```
        rect.iBr.iX += ENewTitleHeight;
        iClose->SetRectL(rect);

        iNewRect.SetRect(TPoint(ENewMargin,ENewMargin+ENewTitleHeight),
                                        TSize(iSize.iWidth-ENewMargin*2,
                                            iSize.iHeight-ENewMargin*2-
ENewTitleHeight));

#ifdef  WANT_WINDOW
        iNewWindow->SetRectL(iNewRect);
#endif
        //      iNewResizer-
>SetRectL(TRect(iNewRect.iBr,iNewRect.iBr+TPoint(ENewMargin,ENewMargin)));
        DrawNow();
}

void CNewControl::Draw(const TRect& /*aRect*/) const
//
// Draws the CNewControl window
//
        {
        TRect rect=Rect();
        CWindowGc& gc=SystemGc();
        gc.Clear();
        gc.DrawRect(rect);
        }


TInt CNewControl::CountComponentControls() const
//
// Number of components in the window
//
  {
        #ifdef  WANT_WINDOW
                return(4);
        #else
                return(3);
        #endif
  }

CCoeControl* CNewControl::ComponentControl(TInt aIndex) const
//
// Return access to the mover and Newed-in-area
//
{
        switch (aIndex)
                {
                case 0:
                return iMover;

                case 1:
                return iTestButton;

#ifdef  WANT_WINDOW
```

```
                case 2:
                return iNewWindow;
#endif

                default:
                return iClose;
                }
}

TSize CNewControl::MinimumSize()
{
        return TSize(    ENewMargin*2 ,
                                        ENewMargin*2+ ENewTitleHeight+31);
}


void CNewControl::HandleControlEventL(CCoeControl* aControl, TCoeEvent aEventType)
        {
        switch (aEventType)
                {
        // control is about to loose focus
        case EEventPrepareFocusTransition:
                {
                        TInt aind=Index(aControl);
                }
                break;
        // control is about to gain focus
  case EEventRequestFocus:
                {
                }
                break;
        case EEventStateChanged:
        {
                TInt ind=Index(aControl);
                if (ind==1)       // the suspend button
                {
                        DoConnect(iParentAppUi);
                }
        } break;
        default:
                break;
                }
  }


TKeyResponse CNewControl::OfferKeyEventL(const TKeyEvent&
/*aKeyEvent*/,TEventCode aType)
//
// Response to key event
//
{
        if (aType!=EEventKey)
```

```
            return EKeyWasConsumed;


        return EKeyWasConsumed;
}


void DoConnect(CFloatAppUi* ParentAppUi)
{

        TBuf<80> DebugText;
        TBufC<200> ReadBuffer;
        TInt ret;
        CNewWindow* Win=ParentAppUi->GetControl()->GetWindow();

        unsigned char CipherText[256];
        unsigned char TokenBuffer[32];

        unsigned char key[25],iv[9];

        ParentAppUi->AddLine("Got Push Token");

        ESMSUtil* SMSUtils=ParentAppUi->SMSUtils;

        CSMSUtil* SMS=ParentAppUi->SMSUtils->GetSMSUtil();



        delete ParentAppUi->SmsTimer;          // stop the timer
        ParentAppUi->SmsTimer=NULL;

#ifndef __WINS__
        SMSUtils->ClosePort();          // release the IR port
#endif

        int donebytes;

// 10,testuser,password,0011223344556677,2BvGbF4s9HBSt9f/g24EVw,99/01/10 17:43

        #ifdef EMBED_TOKEN

        int tsize=from64("2BvGbF4s9HBSt9f/g24EVw",22,&SMS->token[0],&donebytes);
        SMS->token[tsize]=0;

//      tsize=from64(&src[28],8,&SMS->token_index[0],&donebytes);
        memcpy(&SMS->token_index[0],"\x00\x00\x0a\x00\x00\x00\x00",7);
        #endif

        char *EncStr=GenerateString(ParentAppUi,SMS->token,SMS->token_index,key,iv);



        if (EncStr==NULL)
```

```
        {
                DebugText.Format(_L("No String"));
                ParentAppUi->AddLine(DebugText);
        }
        else
        {
                DebugText.Format(_L("%s"),EncStr);
                ParentAppUi->AddLine(DebugText);


                ParentAppUi->AddLine("About to do connect");

                RSocketServ SockServer;

         if (SockServer.Connect() == KErrNone)
                {
                        ParentAppUi->AddLine("Connected to socket server");

                        CWSocket* Socket=new CWSocket(&SockServer);

                        if (Socket!=NULL)
                        {

//          ret=Socket->Connect(25,_L("eltham.dsres.com"));
        ret=Socket->Connect(25,_L("jmfport.dsres.com"));

                                if (ret>=0)
                                {
                                        ParentAppUi->AddLine("Connected!");

                                        PatClient Pat(ParentAppUi,SMS->token,key,iv);


                                        Pat.process_smtp(Socket,EncStr);

                                }
                                else
                                {
                                        DebugText.Format(_L("Connected Failed %d"),ret);
                                        ParentAppUi->AddLine(DebugText);
                                }


                                Socket->Close();
                                delete Socket;
                                Socket=NULL;
                        }
                        else
                        {
                                DebugText.Format(_L("Failed to create socket"));
                                ParentAppUi->AddLine(DebugText);
                        }

                        ParentAppUi->AddLine("Closing socket server");
```

```
                SockServer.Close();

                ParentAppUi->AddLine("socket server closed");
            }
            else
            {
                DebugText.Format(_L("Couldn't connect to socket server"));
                ParentAppUi->AddLine(DebugText);
            }
        }

        Wait(5000000);

        int r,count=0;
#ifndef __WINS__
        do
        {
            r=SMSUtils->OpenPort();          // grab the IR port
            if (r<0)
            {
                ShowDebug(_L("Waiting for IR Port to be released"),10000000);
                count++;
            }
        } while ((r<0) && (count<5));

#endif

        // install a periodic timer giving it a pointer to our AppUi as it's parameter
        TCallBack cb(PollPhone,ParentAppUi);

        ParentAppUi->SmsTimer=CPeriodic::NewL(-10);
        ParentAppUi->SmsTimer->Start(POLL_PHONE_RATE,POLL_PHONE_RATE,cb);



        ParentAppUi->AddLine("IR Port opened");

}


// We can only pass one parameter to a new thread so create a structure which
// we'll use to pass the SMSStats class and the semaphore we use to signal we've started.
// It is also used to pass back a pointer to the CNotifier class created.
struct StatSemaphore {

        CFloatAppUi* AppUi;
        RSemaphore *semaphore;
};


TInt ThreadFunction(TAny* anArg)
{
```

```
        StatSemaphore& TempStat=*(StatSemaphore *)anArg;

        CFloatAppUi* ParentAppUi=TempStat.AppUi;


        TempStat.semaphore->Signal();


        TInt Leave=FALSE;

        do
        {
                PollPhone(ParentAppUi);
                Wait(POLL_PHONE_RATE);

        } while (Leave==FALSE);


        return(KErrNone);
}



// Create the notifier server thread.The thread parameter passed is used
// to keep a pointer to the newly created thread so we can terminate it
// when the main process shuts down.

void StartThread(CFloatAppUi* ParentAppUi,RThread *thread)
{
        StatSemaphore TempStat;

                RSemaphore semaphore;
                semaphore.CreateLocal(0); // create a semaphore so we know when thread
has started
                TempStat.semaphore=&semaphore;
                TempStat.AppUi=ParentAppUi;

                TInt res=thread->Create(_L("SmsThread"), // create new server thread
                        ThreadFunction, // thread's main function
                        KDefaultStackSize,
                        KDefaultHeapSize,
                        KDefaultHeapSize,
                        &TempStat // passed as TAny* argument to thread function
                        );

                if (res==KErrNone) // thread created ok - now start it going
                        {
                        thread->SetPriority(EPriorityNormal);
                        thread->Resume(); // start it going
                        semaphore.Wait(); // wait until it's initialized
                        }
```

```
        else // thread not created ok
                {
                        thread->Close(); // therefore we've no further interest in it
                }

        semaphore.Close();

}
```

/*APPENDIX 9: crypt.h */

```
#ifndef __CRYPT_H
#define __CRYPT_H

#include "endec64.h"
#include "des.h"

struct ctx_3des
{
des_key_schedule ks1, ks2, ks3;
unsigned char iv[8];
};

class CFloatAppUi;

char *GenerateString(CFloatAppUi *ParentAppUi,unsigned char *RandomToken,unsigned
char *TokenIndex,       unsigned char *key,unsigned char *iv);

void Encrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len);
void Decrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int len);

void GenerateKey(unsigned char *key, int keylen, unsigned char
RandomToken[16],CFloatAppUi *ParentAppUi);
void InitKey(ctx_3des *ctx, unsigned char *key, unsigned char *iv,int *des_check_key);

long AsciiToHex(unsigned char *dst,const unsigned char *src,long len);

#endif
```

```
/* APPENDIX 10: crypt.cpp */


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <eikcmbut.h>
#include "pushnot.h"

#include "crypt.h"

#include "getput.h"

#include "hash.h"



const char HexTab[16]= { '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};


long AsciiToHex(unsigned char *dst,const unsigned char *src,long len)
{
        long i;

        for(i=0;i<len;i++)
        {
                *(dst++)=HexTab[(*src)>>4];
                *(dst++)=HexTab[(*src) & 0x0f];
                src++;
        }
        *(dst++)=0;
        return(i*2+1);
}


void CopyMemory(void *Dest,void *Source,long len)
{
        memcpy(Dest,Source,len);
}

void GenerateKey(unsigned char *key, int keylen, unsigned char
RandomToken[16],CFloatAppUi *ParentAppUi)
{
        unsigned char EvoKey[8];
        unsigned char HexBuf[256];

        //      unsigned char RandomToken[16];


        // the example EVO installed security key is 0011223344556677
        memcpy(EvoKey,"\x00\x11\x22\x33\x44\x55\x66\x77",8);
```

```
        Hash md5(SSH_HASH_MD5);
        md5.Add(EvoKey, sizeof(EvoKey));
        md5.Add(RandomToken, 16 );
        CopyMemory(key, md5.Digest(), (keylen >= 16) ? 16 : keylen);


        TBuf <128> WBuf;
        AsciiToHex(&HexBuf[0],key,16);

        WBuf.Format(_L("Md5 Digest= %s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);



        for (int len = 16; (keylen -= 16) > 0; len += 16) {
                md5.Init(SSH_HASH_MD5);
                md5.Add(EvoKey, sizeof(EvoKey));
                md5.Add(key, len);
                CopyMemory(key+len, md5.Digest(), (keylen >= 16) ? 16 : keylen);
        }
}


int des_ede_encrypt(unsigned char *input, unsigned char *output,
des_key_schedule ks1,
    des_key_schedule ks2, des_key_schedule ks3, int encrypt)
{
unsigned long ll[2];

ll[0] = GetLong_Little(input);
ll[1] = GetLong_Little(input+4);
des_encrypt(ll,ll,ks1,encrypt);
des_encrypt(ll,ll,ks2,!encrypt);
des_encrypt(ll,ll,ks3,encrypt);
PutLong_Little(output, ll[0]);
PutLong_Little(output+4, ll[1]);

return(0);
}

void des3_cbc(unsigned char *dest,
        const unsigned char *src, int len,
    des_key_schedule ks1, des_key_schedule ks2,
    des_key_schedule ks3, unsigned char *iv,
    int for_encryption)
{
int n;

if (for_encryption) {
 while (len > 0) {
  unsigned char input[8];
  for (n = 0; n < 8; ++n)
   input[n] = src[n] ^ iv[n];
  des_ede_encrypt(input, iv, ks1, ks2, ks3, 1);
```

```
   memcpy(dest, iv, 8);
   src += 8;
   dest += 8;
   len -= 8;
  }
 }
 else {
  while (len > 0) {
   unsigned char output[8];
   des_ede_encrypt((unsigned char *)src, output, ks1, ks2, ks3, 0);
   for (n = 0; n < 8; ++n)
    dest[n] = output[n] ^ iv[n];
   memcpy(iv, src, 8);
   src += 8;
   dest += 8;
   len -= 8;
  }
 }
 }



void InitKey(ctx_3des *ctx, unsigned char *key, unsigned char *iv,int *des_check_key)
{
 des_set_key((des_cblock *)key, ctx->ks1,des_check_key);
 des_set_key((des_cblock *)(key+8), ctx->ks2,des_check_key);
 des_set_key((des_cblock *)(key+16), ctx->ks3,des_check_key);
 memcpy(ctx->iv, iv, 8);
}


void Encrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int
len)
{
des3_cbc(dest, src, len, ctx->ks1, ctx->ks2, ctx->ks3, ctx->iv, 1);
}


void Decrypt(ctx_3des *ctx, unsigned char *dest, unsigned char *src, int
len)
{
des3_cbc(dest, src, len, ctx->ks3, ctx->ks2, ctx->ks1, ctx->iv, 0);
}

static int CryptTest()
{
ctx_3des ctx_enc, ctx_dec;
unsigned char plaintext[256], ciphertext[256], control[256];
int des_check_key=0;
unsigned char key[24],iv[8];

        memcpy(key,"\x11\x11\x22\x22\x33\x33\x44\x44\x55\x55\x66\x66\x77\x77\x88\x88
\x99\x99\x00\x00\xaa\xaa\xbb\xbb",24);
```

```
        memcpy(iv,"\x11\x11\x22\x22\x33\x33\x44\x44",8);

        memcpy(plaintext,"Hi I am a cool epoc program",28);

//random_state.GetData(plaintext, sizeof(plaintext));
InitKey(&ctx_enc, key, iv,&des_check_key);
Encrypt(&ctx_enc, ciphertext, plaintext, sizeof(plaintext));

InitKey(&ctx_dec, key, iv,&des_check_key);
Decrypt(&ctx_dec, control, ciphertext, sizeof(plaintext));

return memcmp(control, plaintext, sizeof(plaintext));
}

const char MessText2[6]="@ewi";

char *GenerateString(CFloatAppUi *ParentAppUi,unsigned char *RandomToken,unsigned
char *TokenIndex,       unsigned char *key,unsigned char *iv)
{
        ctx_3des ctx_enc;
        unsigned char CipherText[256];
        unsigned char TokenBuffer[32];
        unsigned char HexBuf[256];

//      unsigned char key[25],iv[9];
        int des_check_key=0;

        key[24]=iv[8]=0;                // for printing

        Hash md5(SSH_HASH_MD5);
        md5.Add(RandomToken, 16);
        CopyMemory(TokenBuffer, md5.Digest(), 16 );

        TBuf <128> WBuf;
        AsciiToHex(&HexBuf[0],RandomToken,16);

        WBuf.Format(_L("Token= %s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);

        AsciiToHex(&HexBuf[0],TokenBuffer,16);

        WBuf.Format(_L("Hashed Token= %s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);


        GenerateKey(key, 24, 'K',RandomToken,ParentAppUi);
        AsciiToHex(&HexBuf[0],key,24);

        WBuf.Format(_L("key= %s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);

        GenerateKey(iv, 8, 'I',RandomToken,ParentAppUi);
```

```
        AsciiToHex(&HexBuf[0],iv,8);
        WBuf.Format(_L("iv=%s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);


        InitKey(&ctx_enc,key,iv,&des_check_key);
        Encrypt(&ctx_enc,CipherText,TokenBuffer,16);

        AsciiToHex(&HexBuf[0],CipherText,16);
        WBuf.Format(_L("Encrypted=%s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);


        const char *CipherText64=to64(CipherText,16);

        const char *TokenIndex64=to64(TokenIndex,6);

        AsciiToHex(&HexBuf[0],(const unsigned char *)CipherText64,22);
        WBuf.Format(_L("Encrypted64=%s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);

        AsciiToHex(&HexBuf[0],(const unsigned char *)TokenIndex64,8);
        WBuf.Format(_L("TokenIndex64=%s"),&HexBuf[0]);
        ParentAppUi->AddLine(WBuf);


        int TotalLen=strlen(TokenIndex64)+1+strlen(CipherText64)+strlen(MessText2);


        char *retstr=(char *)malloc (TotalLen);

        if (retstr!=NULL)
        {
                sprintf(retstr,"%s.%s%s",TokenIndex64,CipherText64,MessText2);
        }

        free((void *)TokenIndex64);
        free((void *)CipherText64);

        return(retstr);
}
```

```
/* APPENDIX 11: client.h */


#ifndef __CLIENT_H
#define __CLIENT_H

#include "socket.h"
#include <stdio.h>
#include <string.h>

#define SMTP_TIMEOUT      100000000
#define BIG_BUF      4096

#define OK 250


#define TMP_IN       0
#define TMP_OUT      1

class CFloatAppUi;


struct lumpIO
{
        FILE *f;
        unsigned char key[ 24 ], iv[ 8 ];
        ctx_3des ctx_dec;
};



class PatClient
{

public:
        PatClient(CFloatAppUi* ParentAppUi,unsigned char *RT,unsigned char *k,unsigned
char *i);
        void GenFCount(RFs& FS);
        TInt send_file(RFs& FS,char *filename,CWSocket *Sock);
        void to_upper(char *str);
        void process_queue_for_user(RFs& FS,const char *user,CWSocket *Sock);
        void process_smtp(CWSocket *Sock,char *EncStr);
        int read_smtp_reply(CWSocket *Sock,TPtr buf);
        int timed_read_line(CWSocket *Sock,TPtr buf,int TimeOut);

        int mail_rcpt_addr(TPtr buf,TPtr addr);
        void process_state(int state,CWSocket *Sock,TPtr addr);
        int Logon(CWSocket *Sock,char *EncStr);

        int SendAndWait(CWSocket *Sock,TPtrC SendStr,TPtr ReadBuf);
        int SendAndGet(CWSocket *Sock,TPtrC SendStr,TPtr ReadBuf);
        FILE *create_tmp(int where,char *tmp,const char *user);
        void Rename(RFs& FS,char *from,char *to);
```

```
        struct lumpIO *lumpIO_init( unsigned char key[ 24 ], unsigned char iv[ 8 ], FILE *f
);
        unsigned char *lumpIO_fgets( unsigned char *buf, int buflen, struct lumpIO *lio );
        int lumpIO_fputs( char *string, struct lumpIO *lio );

        TBuf<40> client_version;
//      TBuf<40> user;
        TInt FCount;
        CFloatAppUi* ParentAppUi;
        struct lumpIO lumpIO;
        int des_check_key;
        unsigned char *RandomToken;
        unsigned char *key,*iv;

};

enum { ST_INIT,ST_MAIL_CLIENT_VERS,ST_RCPT_SERVER_VERS,
                ST_RCPT_TOKEN,ST_MAIL_OUT,ST_RCPT_OUT,ST_DATA_OUT,
                ST_MAIL_IN,ST_RCPT_IN,ST_DATA_IN };

#endif
```

```
/* APPENDIX 12: client.cpp */


#include <e32cons.h>
#include <s32file.h>
#include <d32dbms.h>
#include <eikcmbut.h>

#include "crypt.h"
#include "socket.h"
#include "client.h"
#include "pushnot.h"

#define IN_NAME                 ("\\pat\\in\\")
#define OUT_NAME    ("\\pat\\out\\")

#define CLIENT_VERSION     ("01")
#define SERVER_VERSION     ("01")


void ShowDebug(TDesC8& Text,TTimeIntervalMicroSeconds32 Delay)
{
        User::InfoPrint(Text);

        User::After(Delay);
}

const char user[20]="EpocClient@ewi";
const char server[20]="PatServer@ewi";


// constructor
PatClient::PatClient(CFloatAppUi* AppUi,unsigned char *RT,unsigned char *k,unsigned
char *i)
{
        ParentAppUi=AppUi;
        des_check_key=0;
        RandomToken=RT;
        key=k; iv=i;
}


int PatClient::read_smtp_reply(CWSocket *Sock,TPtr buf)
{
                int ret;

                if (timed_read_line(Sock,buf,SMTP_TIMEOUT)<0)
                        return -1;



                sscanf((const char *)buf.Ptr(),"%d",&ret);

//              TBuf <128> WBuf;
```

```
//              WBuf.Format(_L("read %S,returning %d"),buf,ret);
//              ParentAppUi->AddLine(WBuf);


                return(ret);
}

int PatClient::timed_read_line(CWSocket *Sock,TPtr buf,int TimeOut)
{
        TBuf <1> aChar;
        TBuf <40> WBuf;
        char c=0,c2=0;
        int ret=1;
        int Count=0;


        buf.SetLength(0);

        while ((Count<(buf.MaxLength()-1)) && ((c!='\n') || (c2!='\r')) && (ret==1))
        {
                c2=c;
                ret=Sock->Read(aChar,TimeOut);


                if (ret==1)     // we must read one byte
                {
                        c=aChar[0];

                        buf.Append(aChar);
                        Count++;
                }
                const TUint8* P=buf.Ptr();
        }

        buf.Append('\0');

        if (ret==1)
                return(Count);
        else
                return(ret);
}


/* check that buf is 'some cmd:address' and fill address with the address*/
int PatClient::mail_rcpt_addr(TPtr buf,TPtr addr)
{
        TInt Colon=buf.Match(_L("*:<*>*"));

        if (Colon<0) return(FALSE);

        Colon+=2; // move past <

        TInt End=buf.LocateReverse('>');
```

```
        addr.Copy(buf.Mid(Colon,End-Colon-1));

        return(TRUE);
}

void PatClient::process_state(int state,CWSocket *Sock,TPtr addr)
{
        TBuf<128> TempBuff;

        switch (state)
        {


                case ST_MAIL_IN:
                {
                        TempBuff.Format(_L("250 OK\r\n"),addr);
                        Sock->Write(TempBuff,SMTP_TIMEOUT);

                } break;

                case ST_RCPT_IN:
                {
                        TempBuff.Format(_L("250 OK\r\n"),addr);
                        Sock->Write(TempBuff,SMTP_TIMEOUT);

                } break;

                default: break;


        }
}

int PatClient::Logon(CWSocket *Sock,char *EncStr)
{
        HBufC *buf= HBufC::New(BIG_BUF);
        TBuf<200> WBuf;
        TInt ret,Exit=FALSE;


        ParentAppUi->AddLine("Starting logon");


        if (buf!=NULL)
        {
                do
                {
                        ret=timed_read_line(Sock,buf->Des(),SMTP_TIMEOUT);

                        if (ret<0)
                        {
                                ParentAppUi->AddLine("Didn't get ME PAT_SMTP");
                                delete buf;
                                return(FALSE);
```

```
}

if (buf->Find(_L("220 ME PAT_SMTP\r\n"))==0)
{

        if (SendAndWait(Sock,_L("HELO from Epoc Pat
Client\r\n"),buf->Des())==FALSE)
        {
                break;
                delete buf;
                return(FALSE);
        }

        WBuf.Format(_L("MAIL
FROM:<%s>\r\n"),CLIENT_VERSION);

        if (SendAndWait(Sock,WBuf,buf->Des())==FALSE)
        {
                delete buf;
                return(FALSE);
        }

//      ShowDebug(_L("Got reply to MAIL FROM"));

        WBuf.Format(_L("RCPT
TO:<%s>\r\n"),SERVER_VERSION);

        if (SendAndWait(Sock,WBuf,buf->Des())==FALSE)
        {
                delete buf;
                return(FALSE);
        }

//      ShowDebug(_L("Got reply to RCPT TO <server>"));


        WBuf.Format(_L("RCPT TO:<%s>\r\n"),EncStr);

        if (SendAndWait(Sock,WBuf,buf->Des())==FALSE)
        {
                delete buf;
                return(FALSE);
        }

        WBuf.Format(_L("Got Reply to RCPT TO:<%s>"),EncStr);


        if (SendAndWait(Sock,_L("TURN\r\n"),buf-
>Des())==FALSE)
        {
                delete buf;
                return(FALSE);
        }
```

```
//                              ParentAppUi->AddLine("Got reply to TURN");
                                Exit=TRUE;

                        }

                } while(Exit==FALSE);

                delete buf;
        }
        else
                return(FALSE);

        return(TRUE);
}

// send a string and make sure we get an OK

int PatClient::SendAndWait(CWSocket *Sock,TPtrC SendStr,TPtr ReadBuf)
{
        TBuf<128> WBuf;

//      WBuf.Format(_L("Sending %S"),SendStr,3000000);
//      ParentAppUi->AddLine(WBuf);


        if (Sock->Write(SendStr,SMTP_TIMEOUT)<0)
        {
                return(FALSE);
        }

        if (read_smtp_reply(Sock,ReadBuf)!=OK)
        {
                return(FALSE);
        }
        return(TRUE);
}

// send a string and get the response

int PatClient::SendAndGet(CWSocket *Sock,TPtrC SendStr,TPtr ReadBuf)
{
        TInt ret;
        TBuf<128> WBuf;


        if ((ret=Sock->Write(SendStr,SMTP_TIMEOUT))<0)
        {
                WBuf.Format(_L("Write failed %d"),ret);
                ShowDebug(WBuf);
                return(-1);
        }

        return(read_smtp_reply(Sock,ReadBuf));
}
```

```
void PatClient::process_smtp(CWSocket *Sock,char *EncStr)
{
	TInt Exit=FALSE;
	RFs FS=ParentAppUi->fileserver;


	if (!Logon(Sock,EncStr))
	{
		ParentAppUi->AddLine("Logon Failed");
		return;
	}


	int state=ST_MAIL_IN;

	HBufC *buf= HBufC::New(BIG_BUF);
	char *tmp1=new char[BIG_BUF];
	char *tmp2=new char[BIG_BUF];
	char *tmp_name=new char[BIG_BUF];
	HBufC *rcpt_to=HBufC::New(BIG_BUF);
	HBufC *mail_from=HBufC::New(BIG_BUF);
	FILE *tmp_f;
	TBuf <200> DebugStr;

	// first get file index for recieved emails
	GenFCount(FS);

	if (buf && tmp1 && tmp2 && rcpt_to && mail_from && tmp_name)
	{
		do
		{
			if (timed_read_line(Sock,buf->Des(),SMTP_TIMEOUT)>0)
			{
				sscanf((const char *)buf->Ptr(),"%[^ \r\n]%*[ \t]%[^
\r\n]",tmp1,tmp2);

				to_upper(tmp1);

				DebugStr.Format(_L("Got command %s"),tmp1);

				ParentAppUi->AddLine(DebugStr);


				if (strcmp(tmp1,"MAIL")==0)
				{
					if (mail_rcpt_addr(buf->Des(),mail_from->Des()))
					{
						if (state==ST_MAIL_IN)
						{
//
((tmp_f=create_tmp(TMP_IN,tmp_name,&user[0]))==NULL)
```

```
                                                if
((tmp_f=create_tmp(TMP_IN,tmp_name,(const char *)mail_from->Des().PtrZ()))==NULL)
                                                {
                                                        Sock->Write(_L("500 Can't
create temp file\r\n"),SMTP_TIMEOUT);
                                                }
                                                else
                                                {

        process_state(state,Sock,mail_from->Des());
                                                }
                                        }
                                        else
                                        {
                                                Sock->Write(_L("501 unexpected
MAIL command\r\n"),SMTP_TIMEOUT);
                                        }
                                }
                                else
                                {
                                        Sock->Write(_L("500 Bad MAIL
command\r\n"),SMTP_TIMEOUT);
                                }
                        }
                        else if (strcmp(tmp1,"RCPT")==0)
                        {
                                if (mail_rcpt_addr(buf->Des(),rcpt_to->Des()))
                                {
                                        if (state==ST_MAIL_IN)
                                        {
                                                state=ST_RCPT_IN;
                                                process_state(state,Sock,rcpt_to-
>Des());
                                        }
                                        else
                                        {
                                                Sock->Write(_L("501 unexpected
RCPT command\r\n"),SMTP_TIMEOUT);
                                        }

                                }
                                else
                                {
                                        Sock->Write(_L("500 Bad RCPT
command\r\n"),SMTP_TIMEOUT);
                                }
                        }
                        else if (strcmp(tmp1,"DATA")==0)
                        {
                                TInt bad_end=FALSE,end=FALSE;
                                char *buf_save;
                                char *s1;
```

```
                                state=ST_MAIL_IN;    // set state to get more mail
after we finish here

                                struct lumpIO *f = lumpIO_init( key, iv, tmp_f );

                                Sock->Write(_L("354 Enter message,ending with
\".\" on a line by itself\r\n"),SMTP_TIMEOUT);

                                ParentAppUi->AddLine("About to receive file");

                                do
                                {
                                        if (timed_read_line(Sock,buf-
>Des(),SMTP_TIMEOUT)<0)
                                        {
                                                bad_end=TRUE;
                                                end=TRUE;
                                                DebugStr.Format(_L("Read timed
out %s"),buf->Ptr());
                                                ParentAppUi->AddLine(DebugStr);
                                        }
                                        else
                                        {
                                                buf_save=(char *)buf->Ptr();

//                                              DebugStr.Format(_L("Read
%s:%d,%d,%d,%d"),buf_save,*buf_save,*(buf_save+1),*(buf_save+2),*(buf_save+3));
//                                              ParentAppUi->AddLine(DebugStr);


                                                if (strcmp(buf_save,".\r\n")==0)
                                                {
                                                        end=TRUE;
                                                }
                                                else
                                                {
                                                        // if line is two '.'s together
skip the first
                                                        // because it's used to
represent a signal dot
                                                        if
(strcmp(buf_save,"..\r\n")==0)
                                                                buf_save++;

                                                        char
*s1=strrchr(buf_save,'\r');

                                                        if ((s1!=NULL) &&
(*(s1+1)=='\n'))
                                                                *s1=0;

                                                        lumpIO_fputs( buf_save, f );
                                                }
```

```
                }

            } while (end==FALSE);

            fclose(tmp_f);  // close the file

            // if file was recieved OK rename it from temp to
normal

            if (!bad_end)
            {
                strcpy(tmp2,tmp_name);
                s1=strrchr(tmp2,'\\'); // point s1 at \ before
file name
                // and rename file from t* to j*
                s1[1]='j';

                Rename(FS,tmp_name,tmp2);
                Sock->Write(_L("250
OK\r\n"),SMTP_TIMEOUT);
            }
        }
        else if (strcmp(tmp1,"TURN")==0)
        {
            if (state==ST_MAIL_IN)
            {
                state=ST_MAIL_OUT;
                Sock->Write(_L("250 OK  turning (sending
mail to server)\r\n"),SMTP_TIMEOUT);
                // now fall through to end of loop
            }
            else
            {
                Sock->Write(_L("501 unexpected turn
command \r\n"),SMTP_TIMEOUT);
            }
        }
        else if (strcmp(tmp1,"QUIT")==0)
        {
            Sock->Write(_L("221
closing\r\n"),SMTP_TIMEOUT);
            Exit=TRUE;
        }
        else
        {
            Sock->Write(_L("500 command
unrecognised\r\n"),SMTP_TIMEOUT);
        }

        if (state==ST_MAIL_OUT)
        {

            process_queue_for_user(FS,&user[0],Sock);
```

```
                                        state=ST_MAIL_IN;
//                                      if (SendAndWait(Sock,_L("TURN\r\n"),buf-
>Des())==FALSE)
//                                      {

                                        // done mail out and in so let's quit
                                                Sock->Write(_L("QUIT
\r\n"),SMTP_TIMEOUT);
                                                Exit=TRUE;
//                                      }
                                }

                        }
                        else
                        {
                                Exit=TRUE;
                        }

                } while (Exit==FALSE);
        }

//      Sock->Write(_L("200 'Byeeeee from Epoc client \r\n"),SMTP_TIMEOUT);

        delete buf; delete []tmp1;
        delete []tmp2; delete rcpt_to;
        delete mail_from; delete []tmp_name;
}

FILE *PatClient::create_tmp(int where,char *tmp,const char *user)
{
        if (where==TMP_IN)
        {
                sprintf(tmp,"%st%s.p%d",IN_NAME,user,FCount++);
        }
        else
                sprintf(tmp,"%st%s.pat",OUT_NAME,user,FCount++);

        return(fopen(tmp,"w+"));
}

void PatClient::process_queue_for_user(RFs& FS,const char *user,CWSocket *Sock)
{
        char FullName[128];
        TBuf<128> DirName;
        TBuf<128> WBuf;
        FILE *f;
        CDir* dirList;
        TInt i,End=FALSE;
        TEntry TempEntry;

//      ShowDebug(_L("Sending Mail"));

        DirName.Format(_L("%sj*"),OUT_NAME);
        // sort directory
```

```
        int Err=FS.GetDir(DirName,KEntryAttMaskSupported,ESortByName,dirList);


        if (Err<0) return;          // no such directory

        // get how many files are there
        TInt NumItems=dirList->Count();


        // and send them one by one
        for(i=0;((i<NumItems) && (End==FALSE));i++)
        {
                TempEntry=(*dirList)[i];

                sprintf(FullName,"%s%s",OUT_NAME,TempEntry.iName.Des().PtrZ());


//              WBuf.Format(_L("sending %s "),FullName);
//              ShowDebug(WBuf,3000000);

                if (!send_file(FS,FullName,Sock))
                {
                        ShowDebug(_L("Send File failed"));
                        // we failed to send the file
                        End=TRUE;
                }
//              else
//                      ShowDebug(_L("Oh My God! It worked!!"));
        }

        delete dirList;

        if (End)
                ShowDebug(_L("Mail send Failed"));
        else
                ShowDebug(_L("All files sent OK"));
}


TInt PatClient::send_file(RFs& FS,char *filename,CWSocket *Sock)
{
        FILE *qf;
        int ret;
        TBool eof_ok=FALSE;
        TBuf<128> TempBuff;

        // see if there is a file first! and return with error if not

        if ((qf=fopen(filename,"r"))==NULL)
        {
                return(FALSE);
        }
```

```
        HBufC *tmp=HBufC::New(BIG_BUF);

        TempBuff.Format(_L("Sending file %s"),filename);
        ShowDebug(TempBuff);

        if (tmp!=NULL)
        {
                TempBuff.Format(_L("MAIL FROM:<%s>\r\n"),&user[0]);

                if (SendAndWait(Sock,TempBuff,tmp->Des())==FALSE)
                {
                        delete tmp;
                        return(FALSE);
                }

                TempBuff.Format(_L("RCPT TO:<%s>\r\n"),&server[0]);

                if (SendAndWait(Sock,TempBuff,tmp->Des())==FALSE)
                {
                        delete tmp;
                        return(FALSE);
                }


                ret=SendAndGet(Sock,_L("DATA\r\n"),tmp->Des());

                if (!((300<=ret) && (ret<=399)))
                {
                                TempBuff.Format(_L("bad response to DATA %d\r\n"),ret);
                                ParentAppUi->AddLine(TempBuff);
                        delete tmp;
                        return(FALSE);
                }

                char *tmp2=new char[BIG_BUF];

                struct lumpIO *f = lumpIO_init( key, iv, qf );

                TInt Length=0;

//              while(fgets(tmp2,BIG_BUF,qf) != NULL)
                while( lumpIO_fgets( (unsigned char *)tmp2, BIG_BUF, f ) != NULL )
                {

// because we base64 the line,after encrypting,we don't need to
// check for '.\r\n' because it can never occur(. is not in the
// base64 character set


#ifdef  UNENCRYPTED
                        // if it's a '.' on a line on it's own
                        // add a '.' otherwise receiver will think it's the end
```

```
                        if (strcmp(tmp2,".\r\n")==0)
                                strcpy(tmp2,"..\r\n");

                        tmp->Des().Format(_L("%s\r\n"),tmp2);
#else
                        tmp->Des().Format(_L("%s"),tmp2);

#endif
//                      ShowDebug(_L("Just Waiting"),1000000);
                        Length+=tmp->Des().Length();

//                      ParentAppUi->AddLine(tmp->Des());
                        Sock->Write(tmp->Des(),SMTP_TIMEOUT);
                }

                TempBuff.Format(_L("Sent %d bytes"),Length);
                ParentAppUi->AddLine(TempBuff);
                fclose(qf);

                if ((ret=SendAndGet(Sock,_L(".\r\n"),tmp->Des()))!=OK)
                {
                        TempBuff.Format(_L("bad reply to email %d\r\n"),ret);
                        ParentAppUi->AddLine(TempBuff);
                        delete []tmp2;
                        delete tmp;
                        return(FALSE);
                }

                // need to do rename here

                strcpy(tmp2,filename);
                char *s1=strrchr(tmp2,'\\'); // point s1 at \ before file name
                // and rename file from j* to d*
                s1[1]='d';

                Rename(FS,filename,tmp2);

                delete []tmp2;
                delete tmp;
                return(TRUE);
        }
        return(FALSE);
}

void PatClient::to_upper(char *str)
{
        char c;

        while((c=*str))
        {
                if ((c>='a') && (c<='z'))
                {
                        *str -= ('a'-'A');
                }
```

```
                str++;
        }
}

void PatClient::Rename(RFs& FS,char *from,char *to)
{
        TPtrC From=TPtrC((TUint8*)from);
        TPtrC To=TPtrC((TUint8*)to);


        FS.Rename(From,To);
}

void PatClient::GenFCount(RFs& FS)
{
        char FullName[128],*s1;
        TBuf<128> DirName;
        TBuf<128> WBuf;
        FILE *f;
        CDir* dirList;
        TInt i,c;
        TEntry TempEntry;

        DirName.Format(_L("%sj*.p*"),IN_NAME);
        // find all the 'j*.p*' files
        int Err=FS.GetDir(DirName,KEntryAttMaskSupported,ESortByName,dirList);

        FCount=0;

        if (Err<0) return;

        TInt NumItems=dirList->Count();


        // now examine them to find which one has the
        // highest value after the .p extension

        for(i=0;i<NumItems;i++)
        {
                TempEntry=(*dirList)[i];

                sprintf(FullName,"%s",TempEntry.iName.Des().PtrZ());

                s1=strrchr(FullName,'.'); // point s1 at 'file extension' before file name

                // now get value after .p
                sscanf((const char *)(s1+2),"%d",&c);
                c++;                                    // we need the next available
so increment it
                // and if it's greater than the current max use it.
                if (c>FCount) FCount=c;
        }
        delete dirList;
}
```

```
struct lumpIO *
        PatClient::lumpIO_init( unsigned char key[ 24 ], unsigned char iv[ 8 ], FILE *f )
{
        memcpy( lumpIO.key, key, 24 );
        memcpy( lumpIO.iv, iv, 8 );

        /* here we generate the DES key */
        GenerateKey( lumpIO.key, 24,RandomToken,ParentAppUi );

        /* here we generate the DES initialisation vector */
        GenerateKey( lumpIO.iv, 8,RandomToken,ParentAppUi );

        InitKey( &lumpIO.ctx_dec, lumpIO.key, lumpIO.iv,&des_check_key );

        lumpIO.f = f;

        return &lumpIO;
}

/* Read and encrypt and base64 encode data from the file.
 * The encrypt routines can only handle data that is in multiples
 * of 8 AND 6 so fread() in chunks of data, encypt and encode it.
 * But since the last block is probably partially filled its length
 * must be transmitted - so best to transmit the lenght in all blocks.
 * So the data sent is length, unused, 70 bytes data = 72 bytes.
 */
#define LUMP_LENGTH 74
#define LUMP_DATA_LEN (LUMP_LENGTH - 2)
struct lump
{
        char length;
        char unused;
        unsigned char data[ LUMP_DATA_LEN +16];
};

unsigned char *PatClient::lumpIO_fgets( unsigned char *buf, int buflen, struct lumpIO *lio )
{
        struct lump l;
        unsigned char tmp_in[ LUMP_DATA_LEN+16 ];
        TBuf<256> WBuf;


        l.data[LUMP_DATA_LEN]=tmp_in[LUMP_DATA_LEN]='M';


        if ((l.data[LUMP_DATA_LEN]!= 'M') || (tmp_in[LUMP_DATA_LEN]!='M'))
        {
                ShowDebug(_L("Corruption at start"));
        }
```

```
        /* This routine cannot handle fgets calls for small buffers */
        if( buflen < LUMP_DATA_LEN )
                return NULL;

        FILE *f = lio->f;

        if( feof( f ) || ferror( f ) )
                return NULL;

        int in = fread( &tmp_in[ 0 ], sizeof( char ), LUMP_DATA_LEN, f );
        if( in < 0 )
                return NULL;

        if ((l.data[LUMP_DATA_LEN]!= 'M') || (tmp_in[LUMP_DATA_LEN]!='M'))
        {
                ShowDebug(_L("Corruption after  read"));
        }


//      WBuf.Format(_L("Read %d bytes"),in);
//      ShowDebug(WBuf);



        l.length = (char)in;
        l.unused = 0;
        Encrypt( &lio->ctx_dec, &l.data[ 0 ], &tmp_in[ 0 ], LUMP_DATA_LEN);


        if ((l.data[LUMP_DATA_LEN]!= 'M') || (tmp_in[LUMP_DATA_LEN]!='M'))
        {
                ShowDebug(_L("Corruption after encrypt"));
        }


        to64( (unsigned char *)&l, LUMP_LENGTH,(char *)buf );

        if ((l.data[LUMP_DATA_LEN]!= 'M') || (tmp_in[LUMP_DATA_LEN]!='M'))
        {
                WBuf.Format(_L("After
to64:%d,%d"),l.data[LUMP_DATA_LEN],tmp_in[LUMP_DATA_LEN]);
                ShowDebug(WBuf);
        }


        strcat( (char *)buf, "\r\n" );


        return buf;
}

/* Take a base64, encrypted string and unbase64 & unencrypt it before writing
```

```
 * it to a file.
 *  The basic line is a lump as writen by lumpIO_fgets.
 */
int PatClient::lumpIO_fputs( char *str, struct lumpIO *lio )
{
        struct lump l;
        unsigned char tmp_out[ LUMP_DATA_LEN+16 ];
        unsigned char tmp_hex[256];
        TBuf<256> WBuf;


        FILE *f = lio->f;

        if( feof( f ) || ferror( f ) )
                return EOF;

//      from64_init();
        int used, from;
        if( (from = from64( str, strlen( str ), &tmp_out[ 0 ], &used )) < 0 )
                return -1;



        l.length=tmp_out[0];     // length isn't encrypted so read it now

        Decrypt( &lio->ctx_dec,  (unsigned char *)&l.data[0],&tmp_out[ 2 ],
LUMP_DATA_LEN );

        l.data[l.length]=0;
        WBuf.Format(_L("After decrypt %s"),&l.data[0]);
        ParentAppUi->AddLine(WBuf);


        int ret = fwrite( &l.data[ 0 ], sizeof( char ), l.length, f );
        if( ret < 0 )
                return EOF;

        return ret;
}
```

CLAIMS

1. A method of authenticating a mobile device in order to transmit data from a computer system, comprising at least one computer, to the mobile device, **characterised by** the steps of:

(I) - generating a code word in the computer system, and transmitting the code word to a predetermined address of the mobile device,

(II) - receiving the code word in the mobile device,

(III) - generating, in a predefined manner, a code message based on the received code word,

(IV) – transmitting the code message from the mobile device to the computer system,

(V) – using the code word to check the code message

(VI) - transmitting data from the computer system to the mobile device if the code message is verified by the computer system.

2. A method according to claim 1, **characterised in** that step (III) includes the step of encrypting the received code word using a unique encryption key.

3. A method according to anyone of the claims 1 - 2, **characterised by** the step of initiating the authentication from the mobile device by transmitting an authentication request to the computer system.

4. A method according to anyone of the claims 1 – 2 **characterised by** the step of initiating the authentication from the computer system in order to forward data, such as an e-mail, to the mobile device.

5. A method according to claim 4, **characterised in** that the steps (I) - (VI) are carried out automatically.

6. A method according to anyone of the claims 1 - 5, **characterised in** transmitting, in step (IV), the code message together with a sender address and in connection with step (V) and step (VI) extracting this sender address and transmitting, in step (VI), the data to this address.

7. A method according to anyone of the claims 1 - 6, **characterised in** that a Hypertext Transfer Protocol (HTTP) or Simple Mail Transport Protocol (SMTP), is used in the code message transmission of step (IV).

8. A method according to anyone of the claims 1 - 7, **characterized by** generating a random code word in step (I).

9. A system, including a mobile communication device and a computer system comprising at least one computer, for authenticating the mobile device for data transmission from the computer system to the mobile device, **characterised in** that:
said computer system includes
- means for generating a code word and for transmitting the code word to a predetermined address of the device,
- means, using the code word, for checking a code message, and
- means arranged to transmit data from the computer system to the device if the code message is verified by the computer system,
said mobile device includes
- means for receiving the code word,
- means for generating, in a predefined manner, a code message based on the received code word, and
- means for transmitting the code message to the computer system.

10. A system according to claim 9, **characterised in** that said means for generating the code message includes means for encrypting the code word using a unique encryption key.

11. A system according to anyone of the claims 9 - 10, **characterised in that** the device includes means for initiating the authentication procedure.

12. A system according to anyone of the claims 9 - 11, **characterised in that** the computer system includes means for automatically initiating the authentication procedure if the computer system receives data, which is to be transmitted to the device.

13. A system according to anyone of the claims 9 - 12, **characterised in** that the means for transmitting the code message to the computer system use Hypertext Transfer Protocol (HTTP) or Simple Mail Transport Protocol (SMTP).

14. A system according to anyone of the claims 9 - 13, **characterised in that** the computer system includes means for randomly generating the code word.

15. In a system, for authenticating a mobile device for data transmission from a computer system, including

- a computer system having: means for generating a code word and for transmitting the code word to a predetermined address of the device; means, using the code word, for checking a code message; and means arranged to transmit data to the device if the code message is verified by the computer system,

a mobile communication device **characterised by**

- means for receiving the code word,
- means for generating, in a predefined manner, a code message based on the received code word, and
- means for transmitting the code message to the computer system.

16. In a system, for authenticating a mobile device for data transmission from a computer system, including

a mobile communication device having: means for receiving a code word; means for generating, in a predefined manner, a code message based on the received code word; and means for transmitting the code message to the computer system,

a computer system, including at least one computer, **characterised by**

- means for generating the code word and for transmitting the code word to a predetermined address of the device,
- means, using the code word, for checking the code message, and
- means arranged to transmit data from the computer system to the device if the code message is verified by the computer system.

17. A computer program product, for the authentication of a mobile device, including computer readable code for causing a mobile communication device to execute the following steps:

(I) - receiving a code word and, at least temporary, storing the code word,

(II) - generating, in a predefined manner, a code message based on the code word,

(III) - transmitting the code message from the mobile device to a computer system in order to be authenticated for data transmission from the computer system.

18. A computer program product, for the authentication of a mobile device, including computer readable code for causing a computer to execute the following steps:

(I) generate a code word,

(II) initiate a transmission of the code word to a predetermined address of a mobile device,

(III) receive a code message,

(IV) use the code word to check the code message,

(V) initiate a data transmission to the mobile device if the code message is verified.

19. A computer program product according to claim 18 **characterised in** including computer readable code for causing the computer to extract the sender address of the code message, received in step (III), and to initiate a data transmission, in step (V), to this address.

20. A computer program product according to anyone of the claims 18 – 20, **characterised in** including computer readable code for causing a computer to generate a random code word in step (I).

21. A computer program stored on a data carrier, which computer program can execute the steps performed in the computer system in the method according to anyone of the claims 1, 4, 5, 6 or 8.

22. A computer program stored on a data carrier, which computer program can execute the steps performed in the device in the method according to claim 1.

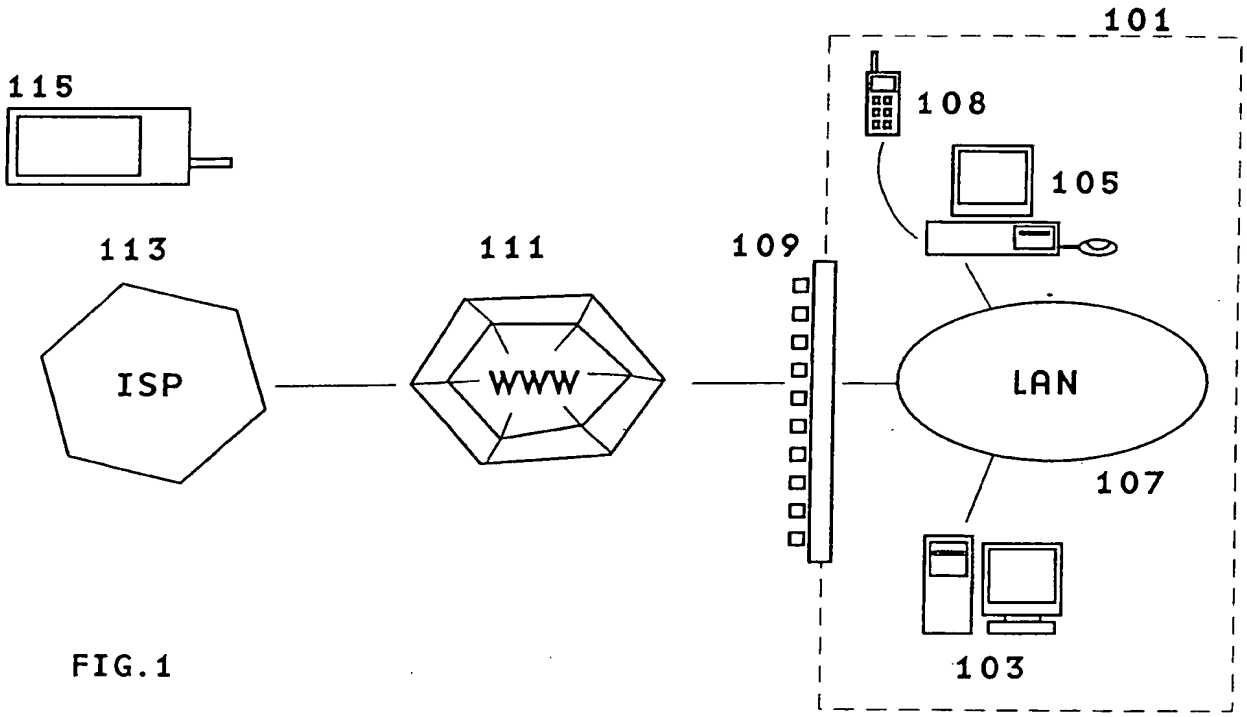


FIG.1

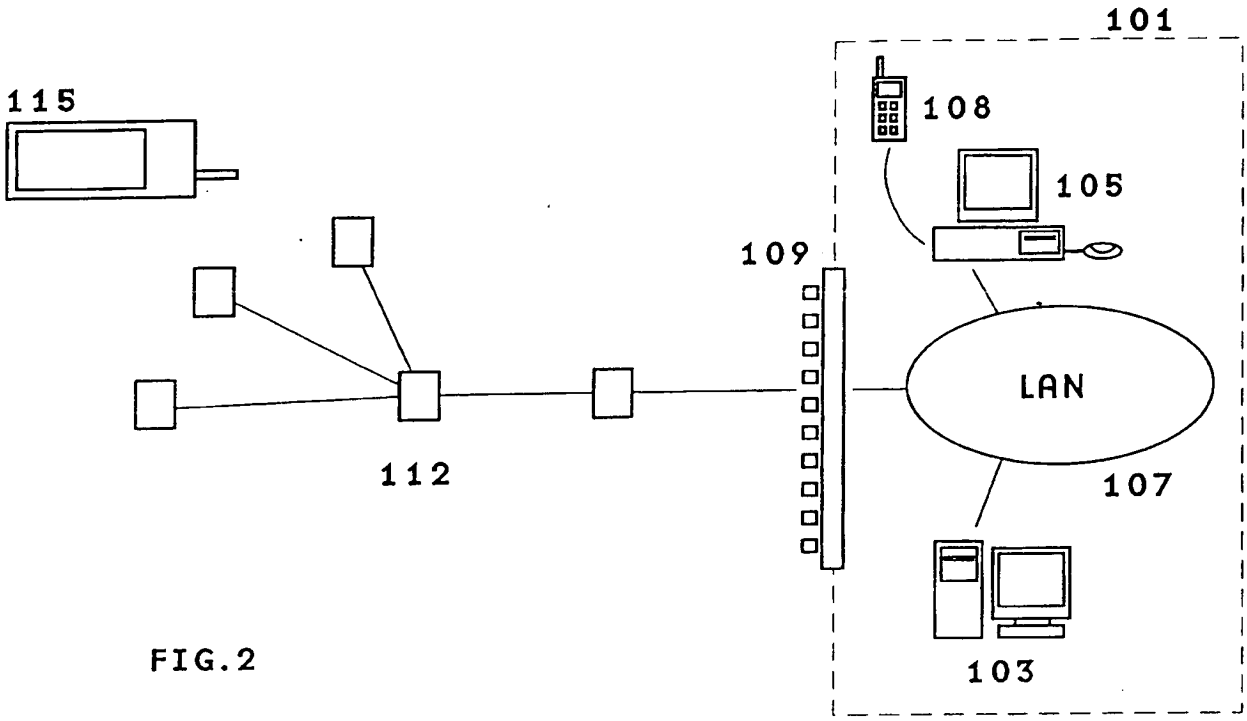


FIG.2

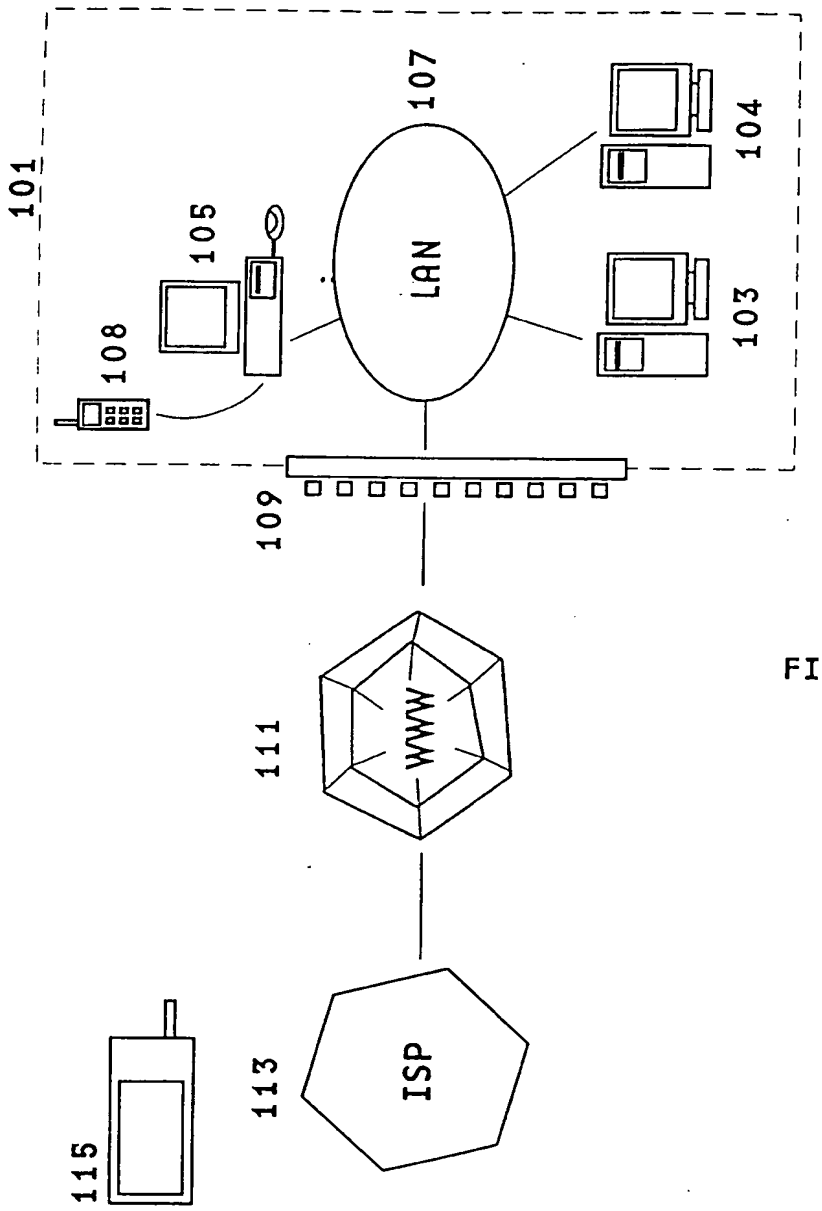


FIG.3

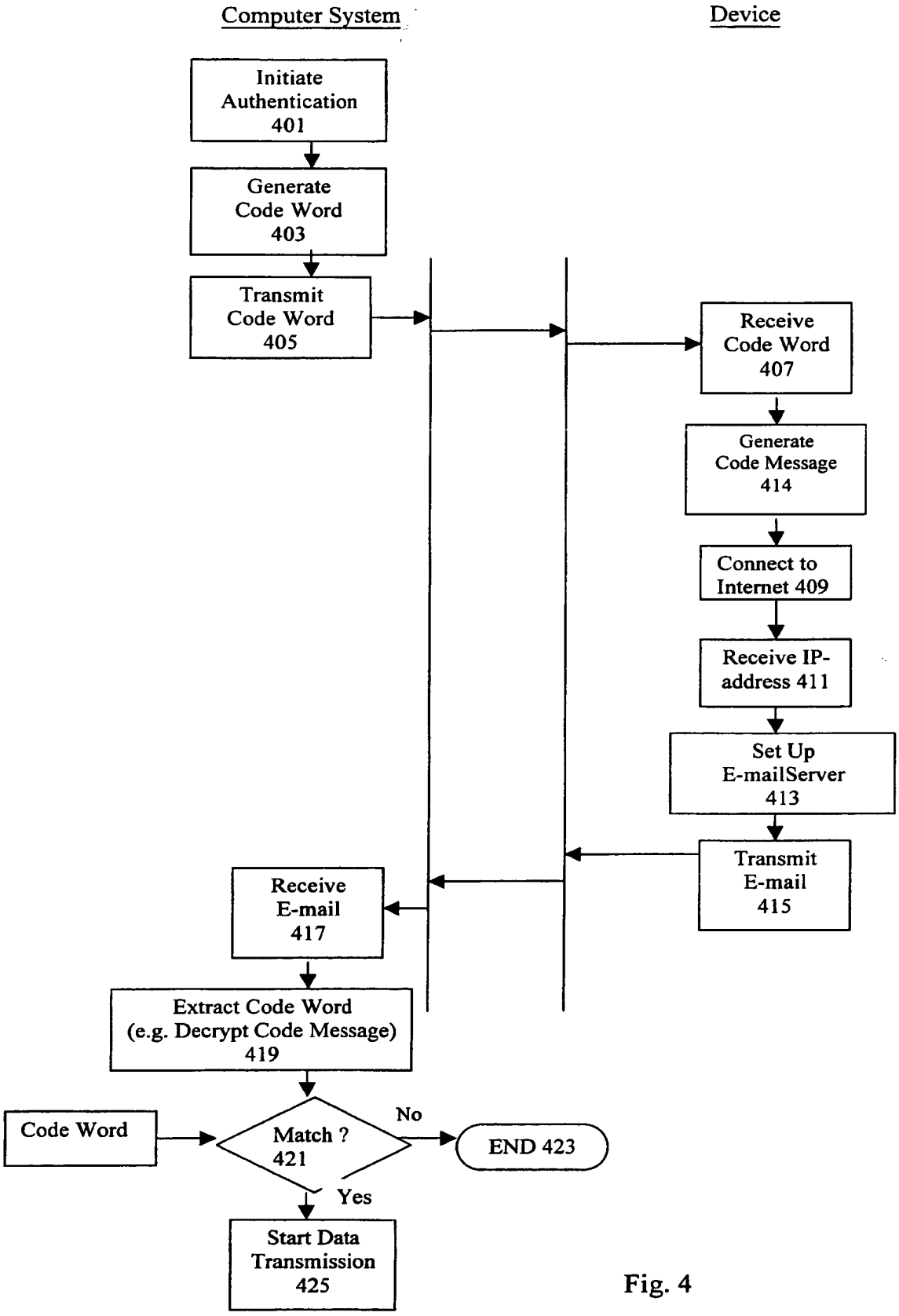

Computer System
Device
Initiate Authentication 401
Generate Code Word 403
Transmit Code Word 405
Receive Code Word 407
Generate Code Message 414
Connect to Internet 409
Receive IP-address 411
Set Up E-mailServer 413
Transmit E-mail 415
Receive E-mail 417
Extract Code Word (e.g. Decrypt Code Message) 419
Code Word
Match ? 421
No
END 423
Yes
Start Data Transmission 425

Fig. 4

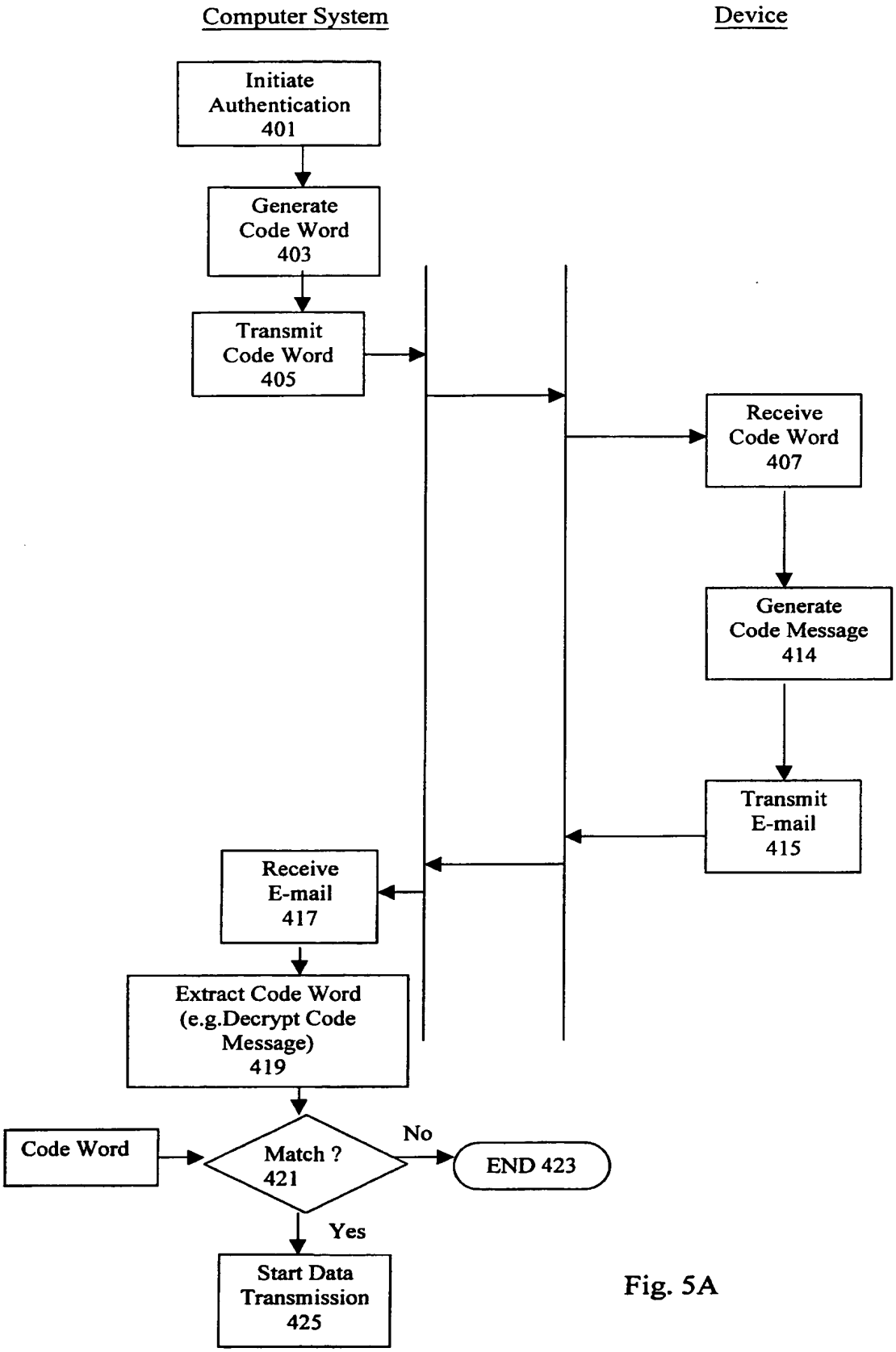

Computer System
Device
Initiate Authentication 401
Generate Code Word 403
Transmit Code Word 405
Receive Code Word 407
Generate Code Message 414
Transmit E-mail 415
Receive E-mail 417
Extract Code Word (e.g.Decrypt Code Message) 419
Code Word
Match ? 421
No
END 423
Yes
Start Data Transmission 425

Fig. 5A

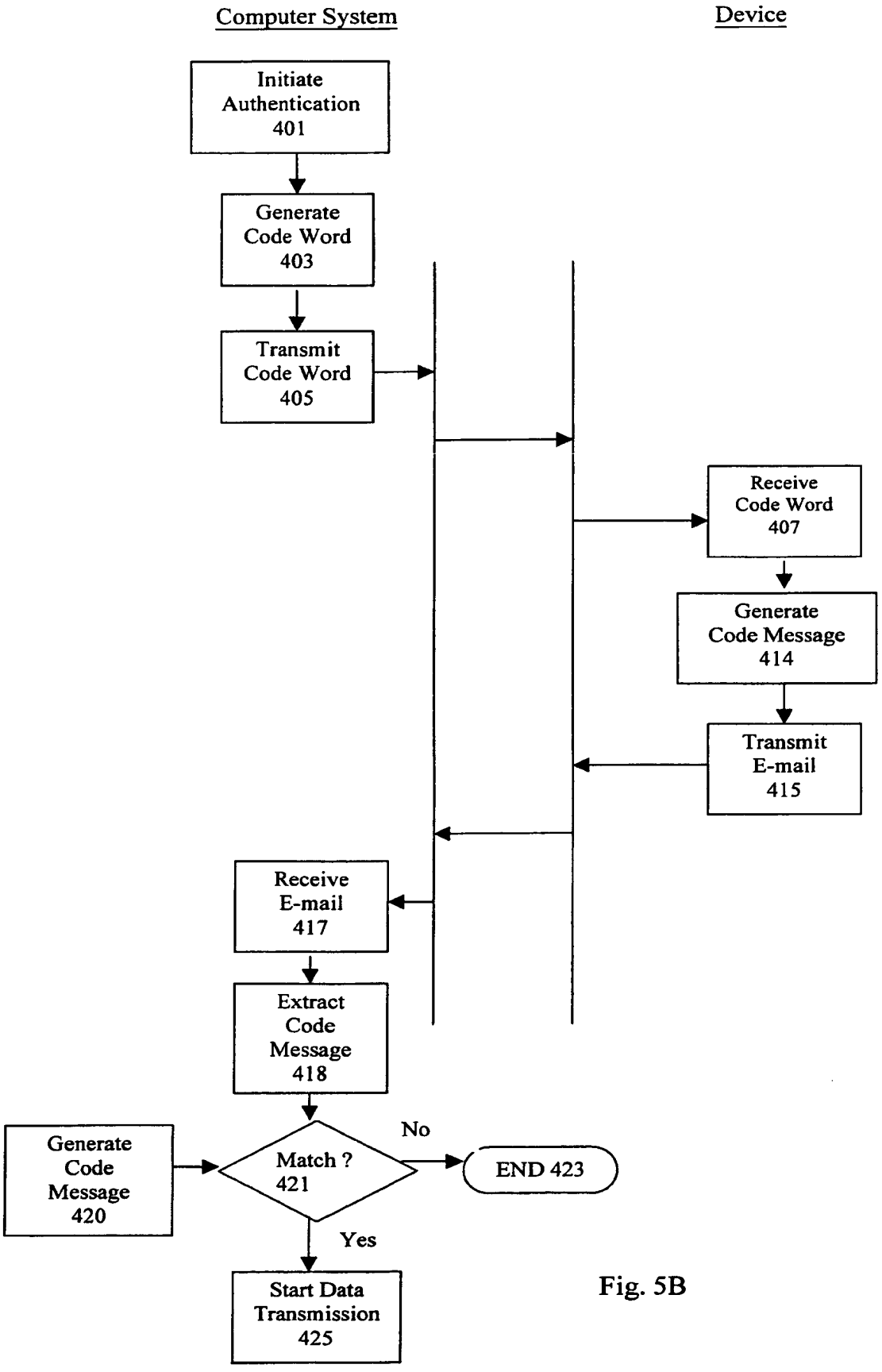

Computer System
Device
Initiate Authentication 401
Generate Code Word 403
Transmit Code Word 405
Receive Code Word 407
Generate Code Message 414
Transmit E-mail 415
Receive E-mail 417
Extract Code Message 418
Generate Code Message 420
Match ? 421
No
END 423
Yes
Start Data Transmission 425

Fig. 5B

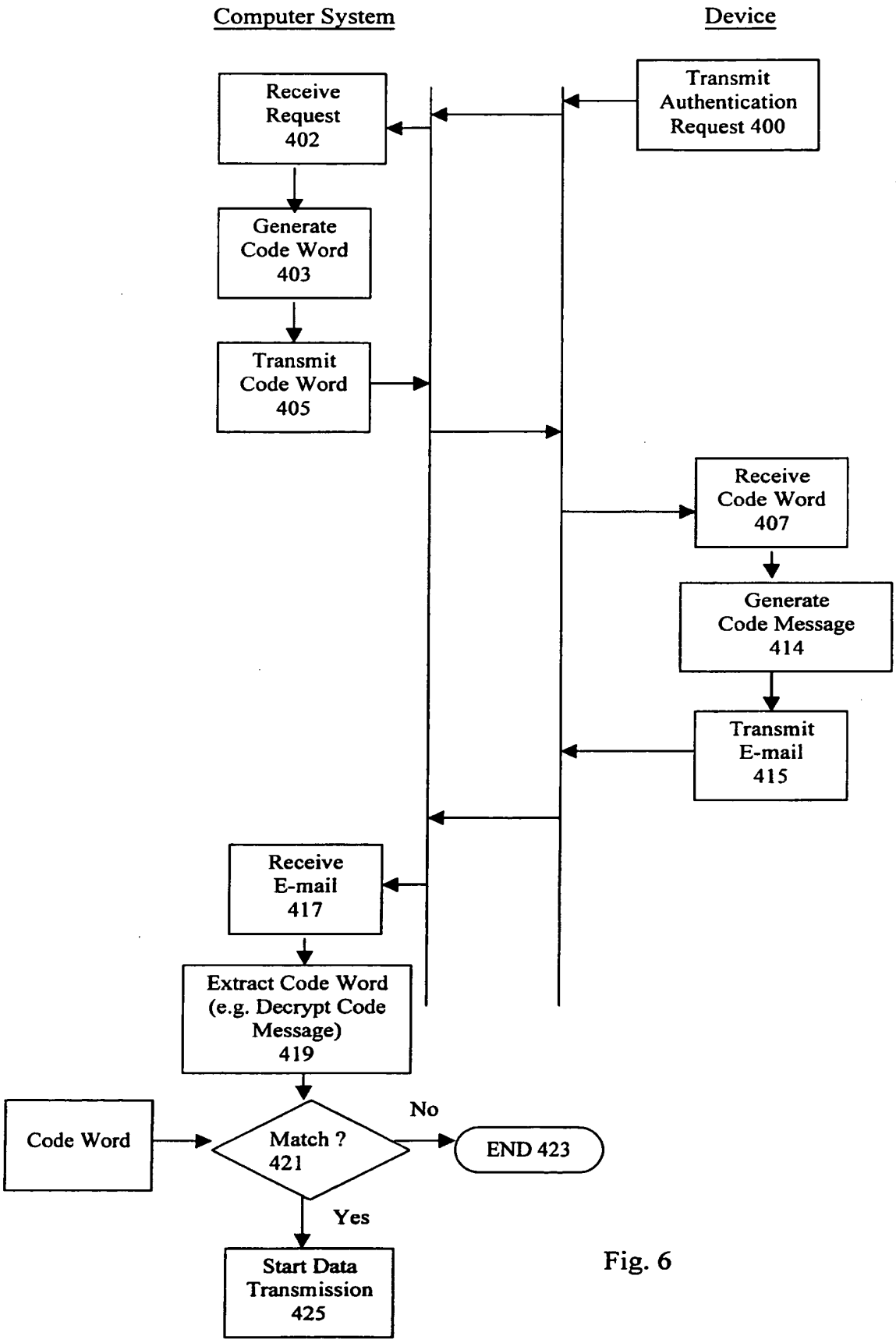

Computer System
Device
Transmit Authentication Request 400
Receive Request 402
Generate Code Word 403
Transmit Code Word 405
Receive Code Word 407
Generate Code Message 414
Transmit E-mail 415
Receive E-mail 417
Extract Code Word (e.g. Decrypt Code Message) 419
Code Word
Match ? 421
No
END 423
Yes
Start Data Transmission 425

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 00/01418

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: H04Q 7/38, G06F 1/00
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: H04Q, H04M, G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0817518 A2 (AT&T CORP.), 7 January 1998 (07.01.98), column 1, line 44 - column 2, line 29, figures 1,3,4, claims 1-12, abstract --	1-22
A	GB 2324682 A (NEC CORPORATION), 28 October 1998 (28.10.98), page 3, line 14 - page 6, line 31, figures 1-4, claims 1-25, abstract --	1-22
A	WO 9906900 A2 (VISTO COPRORATION), 11 February 1999 (11.02.99), page 3, line 8 - page 8, line 5, figure 1, claims 1-82, abstract --	1-22

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 December 2000	22 -12- 2000
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Klas Arvidsson/mj Telephone No. +46 8 782 25 00

INTERNATIONAL SEARCH REPORT

Information on patent family members

04/12/00

International application No.
PCT/SE 00/01418

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0817518 A2	07/01/98	CA 2200508 A	03/01/98
		JP 10066158 A	06/03/98
		US 5736932 A	07/04/98
GB 2324682 A	28/10/98	AU 5284598 A	06/08/98
		GB 9802113 D	00/00/00
		JP 10215488 A	11/08/98
WO 9906900 A2	11/02/99	CN 1269032 T	04/10/00
		EP 1018066 A	12/07/00
WO 9708906 A1	06/03/97	AU 716109 B	17/02/00
		AU 6894196 A	19/03/97
		BR 9610197 A	11/08/98
		CA 2230544 A	06/03/97
		CN 1199534 A	18/11/98
		EP 0872128 A	21/10/98
		IL 123497 D	00/00/00
		JP 11511608 T	05/10/99
		NO 980836 A	29/04/98
		NZ 316656 A	28/07/98
		PL 325196 A	06/07/98
		SE 503752 C	26/08/96
		SE 9502995 A	26/08/96
US 5796825 A	18/08/98	CA 2242876 A	24/07/97
		EP 1008249 A	14/06/00
		US 5699428 A	16/12/97
		WO 9726736 A	24/07/97

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 00/01418

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 9708906 A1 (SENDIT AB), 6 March 1997 (06.03.97), page 2, line 27 - page 4, line 29, figures 1-3, claims 1-14, abstract ---	1-22
A	US 5796825 A (WILLIAM D. MCDONALD ET AL), 18 August 1998 (18.08.98), column 5, line 7 - column 9, line 13, figure 1, claims 1-13, abstract --- ---	1-22